DATA MINING LECTURE 5

MinHashing,

Locality Sensitive Hashing,

Clustering

SKETCHING AND LOCALITY SENSITIVE HASHING

Thanks to:

Rajaraman and Ullman, “Mining Massive Datasets”

Evimaria Terzi, slides for Data Mining Course.

Finding similar documents

- Problem: Find similar documents from a web crawl
- Main issues to address:
  - What is the right representation and similarity function?
    - Shingling: reduce a document to a set of shingles
    - Similarity function: Jaccard similarity
    - $Sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$
  - Compress the sets so that we can fit more in memory
    - Min-Hashing: create a signature/sketch of a document
  - Find similar pairs without checking all possible pairs
    - Locality Sensitive Hashing (LSH)

Shingling

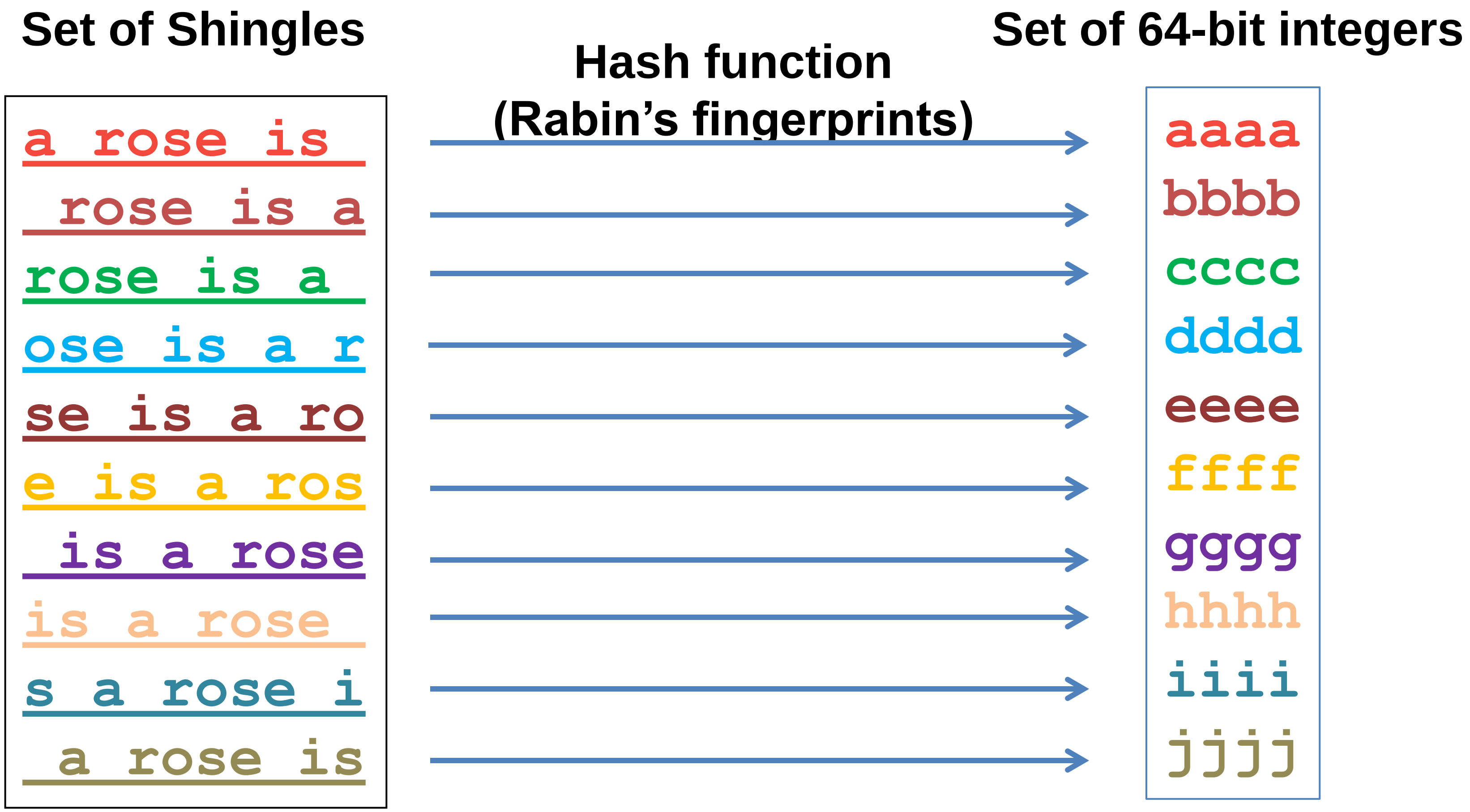
- Shingle: a sequence of k contiguous characters

```
a rose is a rose is a rose
a rose is 
 rose is a
  rose is a 
   ose is a r
    se is a ro
     e is a ros
      is a rose
       is a rose 
        s a rose i
         a rose is
          a rose is 
```

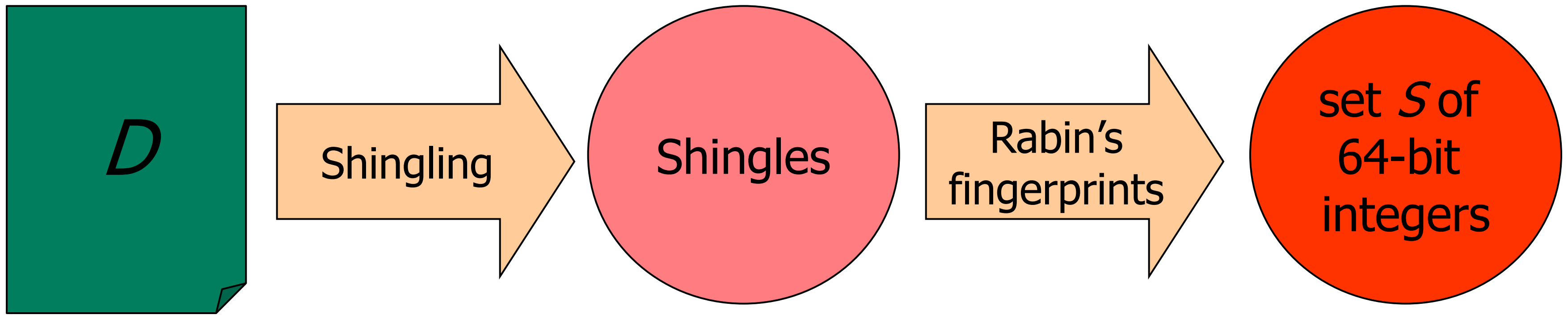
Represent a document as a set of shingles

Fingerprinting

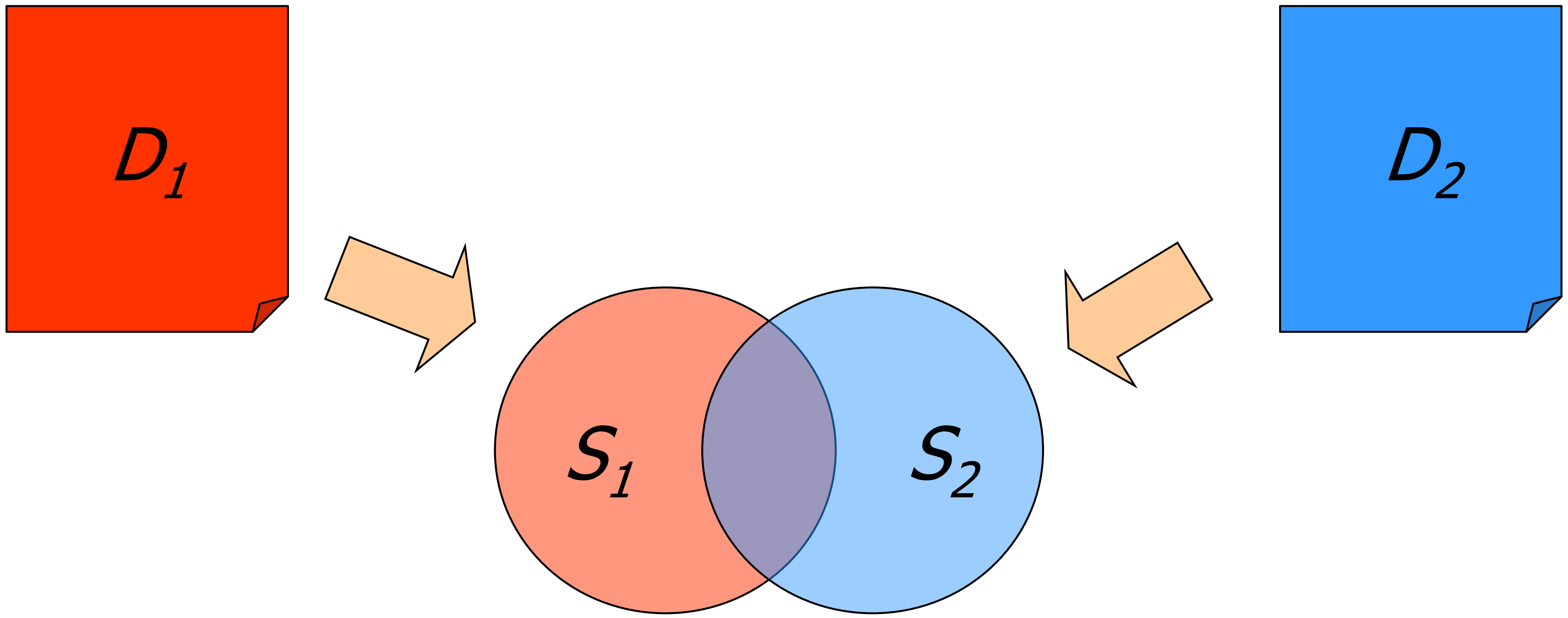
- Hash shingles to 64-bit integers



Document processing



Document Similarity



$$\mathrm{Sim}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Jaccard coefficient

Compacting the data

- Problem: shingle sets are too large to be kept in memory.
- Key idea: “hash” each set S to a small signature Sig (S), such that:
  1. Sig (S) is small enough that we can fit a signature in main memory for each set.
  2. Sim (S_1, S_2) is (almost) the same as the “similarity” of Sig (S_1) and Sig (S_2). (signature preserves similarity).
- Warning: This method can produce false negatives, and false positives (if an additional check is not made).

From Sets to Boolean Matrices

- Represent the data as a boolean matrix M
  - Rows = the universe of all possible set elements
    - In our case, shingle fingerprints take values in [0…2^{64}-1]
  - Columns = the sets
    - In our case, the sets of shingle fingerprints
  - M(e,S) = 1 in row e and column S if and only if e is a member of S.
- Typical matrix is sparse.
  - We do not really materialize the matrix

Example

- Universe: **U = {A,B,C,D,E,F,G}**

- X = {A,B,F,G}
- Y = {A,E,F,G}

- Sim(X,Y) = $\frac{3}{5}$

	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1

Example

- Universe: **U = {A,B,C,D,E,F,G}**

- X = {A,B,F,G}
- Y = {A,E,F,G}

- Sim(X,Y) = $\frac{3}{5}$

At least one of the columns has value 1

	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1

Example

- Universe: **U = {A,B,C,D,E,F,G}**

- X = {A,B,F,G}
- Y = {A,E,F,G}

- $\mathrm{Sim}(X,Y) = \frac{3}{5}$

Both columns have value 1

	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1

Minhashing

- Pick a random permutation of the rows (the universe U).
- Define "hash" function
  - h(S) = the index of the first row (in the permuted order) in which column S has 1.
  - h(S) = the index of the first element of S in the permuted order.
- Use k (e.g., k = 100) independent random permutations to create a signature.

Example of minhash signatures

- Input matrix

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	1	1
F	1	0	1	0
G	1	0	1	0

A
C
G
F
B
E
D

	S_1	S_2	S_3	S_4
A	1	0	1	0
C	0	1	0	1
G	1	0	1	0
F	1	0	1	0
B	1	0	0	1
E	0	1	1	1
D	0	1	0	1

1	**2**	**1**	**2**

Example of minhash signatures

- Input matrix

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	1	1
F	1	0	1	0
G	1	0	1	0

D
B
A
C
F
G
E

	S_1	S_2	S_3	S_4
D	0	1	0	1
B	1	0	0	1
A	1	0	1	0
C	0	1	0	1
F	1	0	1	0
G	1	0	1	0
E	0	1	1	1

2	**1**	**3**	**1**

Example of minhash signatures

- Input matrix

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	1	1
F	1	0	1	0
G	1	0	1	0

C
D
G
F
A
B
E

	S_1	S_2	S_3	S_4
C	0	1	0	1
D	0	1	0	1
G	1	0	1	0
F	1	0	1	0
A	1	0	1	0
B	1	0	0	1
E	0	1	1	1

3	**1**	**3**	**1**

Example of minhash signatures

- Input matrix

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	1	1
F	1	0	1	0
G	1	0	1	0

$\approx$

Signature matrix

	S_1	S_2	S_3	S_4
h_1	1	2	1	2
h_1	2	1	3	1
h_1	3	1	3	1

Sig(S,i) = value of the i-th hash function for set S

Hash function Property

$$\Pr(h(S_1) = h(S_2)) = \mathrm{Sim}(S_1,S_2)$$

- where the probability is over all choices of permutations.
- Why?
  - The first row where one of the two sets has value 1 belongs to the union.
  - We have equality if both columns have value 1.

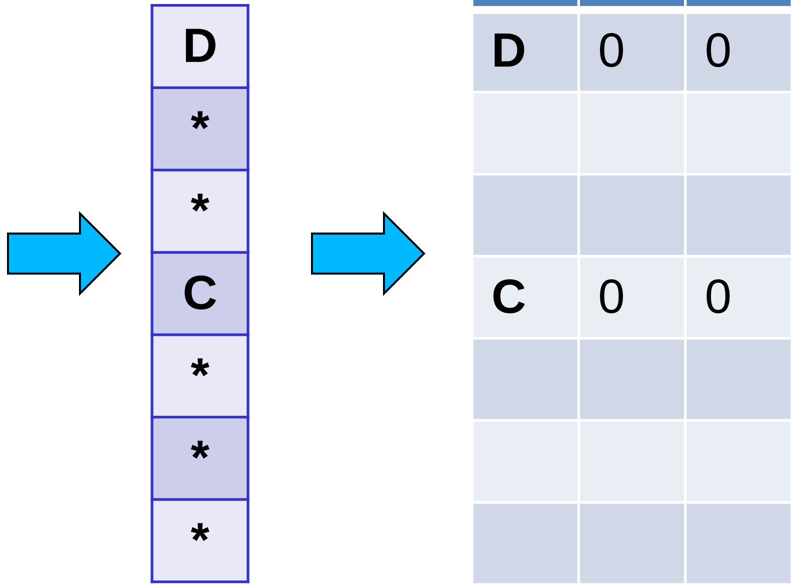
Example

- Universe: **U = {A,B,C,D,E,F,G}**
- X = {A,B,F,G}
- Y = {A,E,F,G}

Rows C,D could be anywhere they do not affect the probability

- Union = {A,B,E,F,G}
- Intersection = {A,F,G}

	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1



D
*
*
C
*
*
*

	X	Y
D	0	0
C	0	0

Example

- Universe: **U = {A,B,C,D,E,F,G}**
- X = {A,B,F,G}
- Y = {A,E,F,G}

The * rows belong to the union

- Union =

  {A,B,E,F,G}
- Intersection =

  {A,F,G}

	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1

D
*****
*****
C
*****
*****
*****

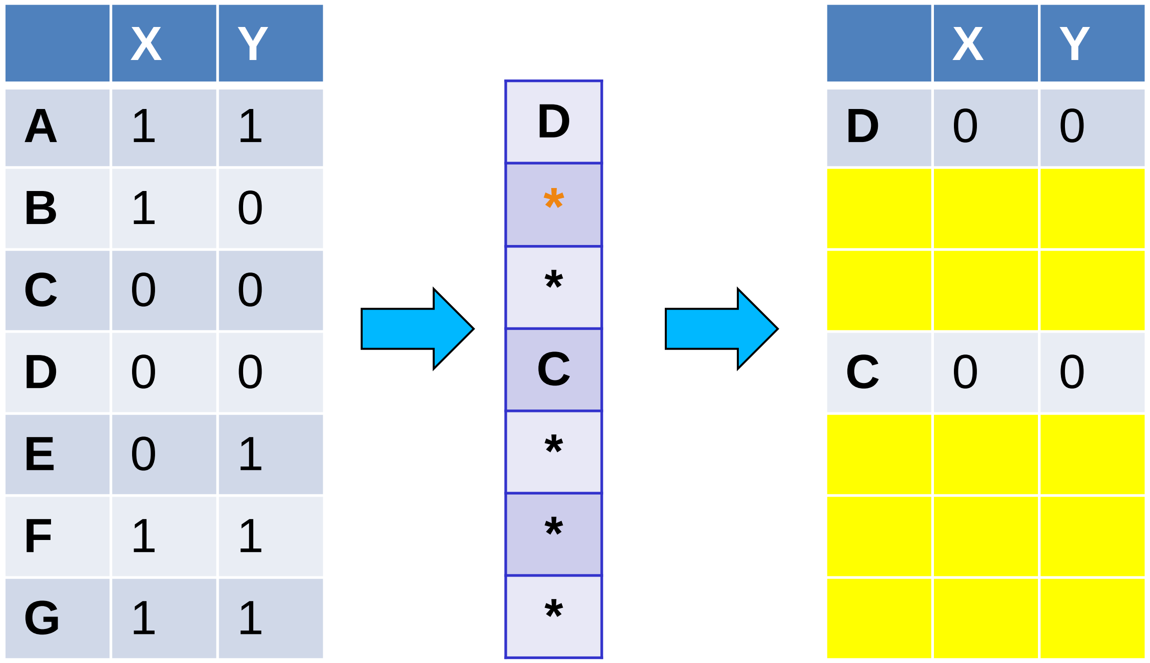
	X	Y
D	0	0
🟨	🟨	🟨
🟨	🟨	🟨
C	0	0
🟨	🟨	🟨
🟨	🟨	🟨
🟨	🟨	🟨

Example

- Universe: **U = {A,B,C,D,E,F,G}**
- X = {A,B,F,G}
- Y = {A,E,F,G}

The question is what is the value of the **first *** element

- Union = {A,B,E,F,G}
- Intersection = {A,F,G}



	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1

D
*****
*****
C
*****
*****
*****

	X	Y
D	0	0
C	0	0

Example

- Universe: **U = {A,B,C,D,E,F,G}**
- X = {A,B,F,G}
- Y = {A,E,F,G}

If it belongs to the intersection then h(X) = h(Y)

- Union =

  {A,B,E,F,G}
- Intersection =

  {A,F,G}

	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1

➡

D
*
*
C
*
*
*

➡

	X	Y
D	0	0
🟥	🟥	🟥
🟨	🟨	🟨
C	0	0
🟨	🟨	🟨
🟨	🟨	🟨
🟨	🟨	🟨

Example

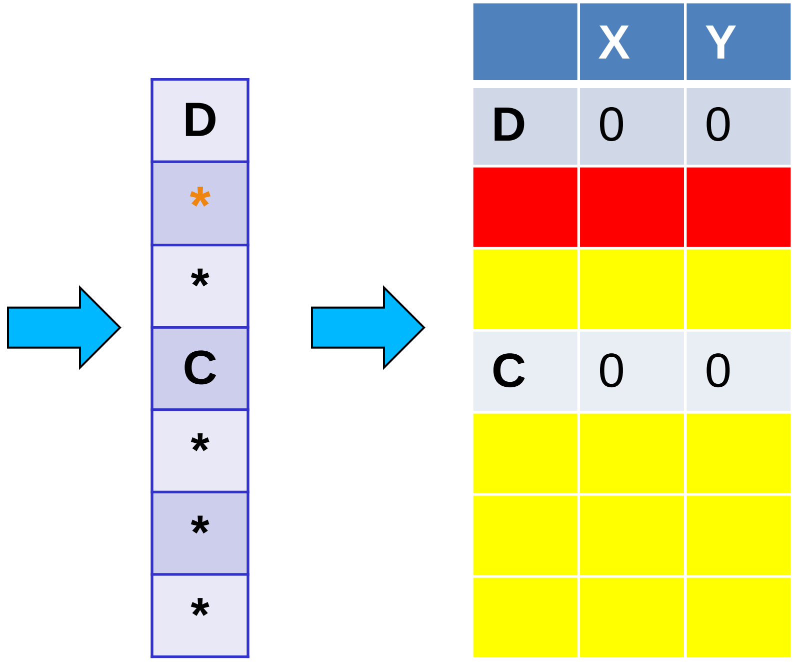
- Universe: **U = {A,B,C,D,E,F,G}**
- X = {A,B,F,G}
- Y = {A,E,F,G}

Every element of the union is equally likely to be the * element

$$\Pr(h(X) = h(X)) = \frac{|\{A,F,G\}|}{|\{A,B,E,F,G\}|} = \frac{3}{5} = \mathrm{Sim}(X,Y)$$

- Union = {A,B,E,F,G}
- Intersection = {A,F,G}

	X	Y
A	1	1
B	1	0
C	0	0
D	0	0
E	0	1
F	1	1
G	1	1



Similarity for Signatures

- The similarity of signatures is the fraction of the hash functions in which they agree.

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	1	1
F	1	0	1	0
G	1	0	1	0

$\approx$

Signature matrix

S_1	S_2	S_3	S_4
1	2	1	2
2	1	3	1
3	1	3	1

	actual	**Sig**
(x1,x2)	0	0
(x1,x3)	3/5	2/3
(x1,x4)	1/7	0
(x2,x3)	0	0
(x2,x4)	3/4	1
(x3,x4)	0	0

- With multiple signatures we get a good approximation

Is it now feasible?

- Assume a billion rows
- Hard to pick a random permutation of 1…billion
- **Even representing a random permutation requires 1 billion entries!!!**
- How about accessing rows in permuted order?
- ☹

Being more practical

- Approximating row permutations: pick **k=100** hash functions **$(h_1,\dots,h_k)$**

```
for each row r
  for each hash function $h_i$
      compute $h_i(r)$
      for each column S that has 1 in row r
         if $h_i(r)$ is a smaller value than Sig(S,i) then
            Sig(S,i) = $h_i(r)$;
```

Sig(S,i) will become the smallest value of **$h_i(r)$** among all rows for which column **S** has value **1**; *i*.e., **$h_i(r)$** gives the min index for the **i**-th permutation

Example

x	Row	S1	S2
0	A	1	0
1	B	0	1
2	C	1	1
3	D	1	0
4	E	0	1

$h(x) = x+1 \bmod 5$
$g(x) = 2x+1 \bmod 5$

Row	S1	S2
E	0	1
A	1	0
B	0	1
C	1	1
D	1	0

Row	S1	S2
B	0	1
E	0	1
C	1	0
A	1	1
D	1	0

	Sig1	Sig2
$h(0) = 1$	1	-
$g(0) = 3$	3	-
$h(1) = 2$	1	2
$g(1) = 0$	3	0
$h(2) = 3$	1	2
$g(2) = 2$	2	0
$h(3) = 4$	1	2
$g(3) = 4$	2	0
$h(4) = 0$	1	0
$g(4) = 1$	2	0

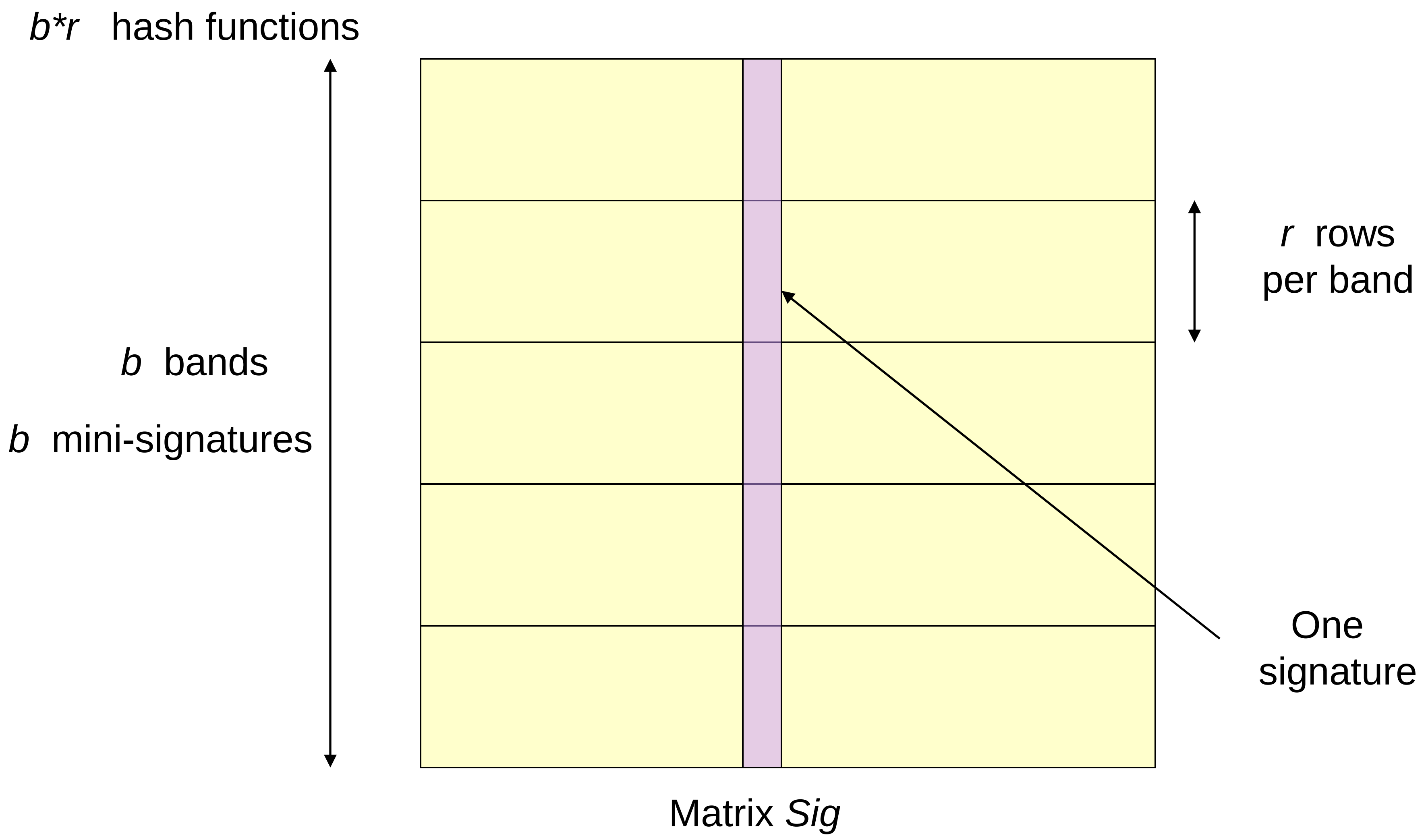
Finding similar pairs

- Problem: Find all pairs of documents with similarity at least t = 0.8
- While the signatures of all columns may fit in main memory, comparing the signatures of all pairs of columns is quadratic in the number of columns.
- Example: 10^6 columns implies $5*10^{11}$ column-comparisons.
- At 1 microsecond/comparison: 6 days.

Locality-Sensitive Hashing

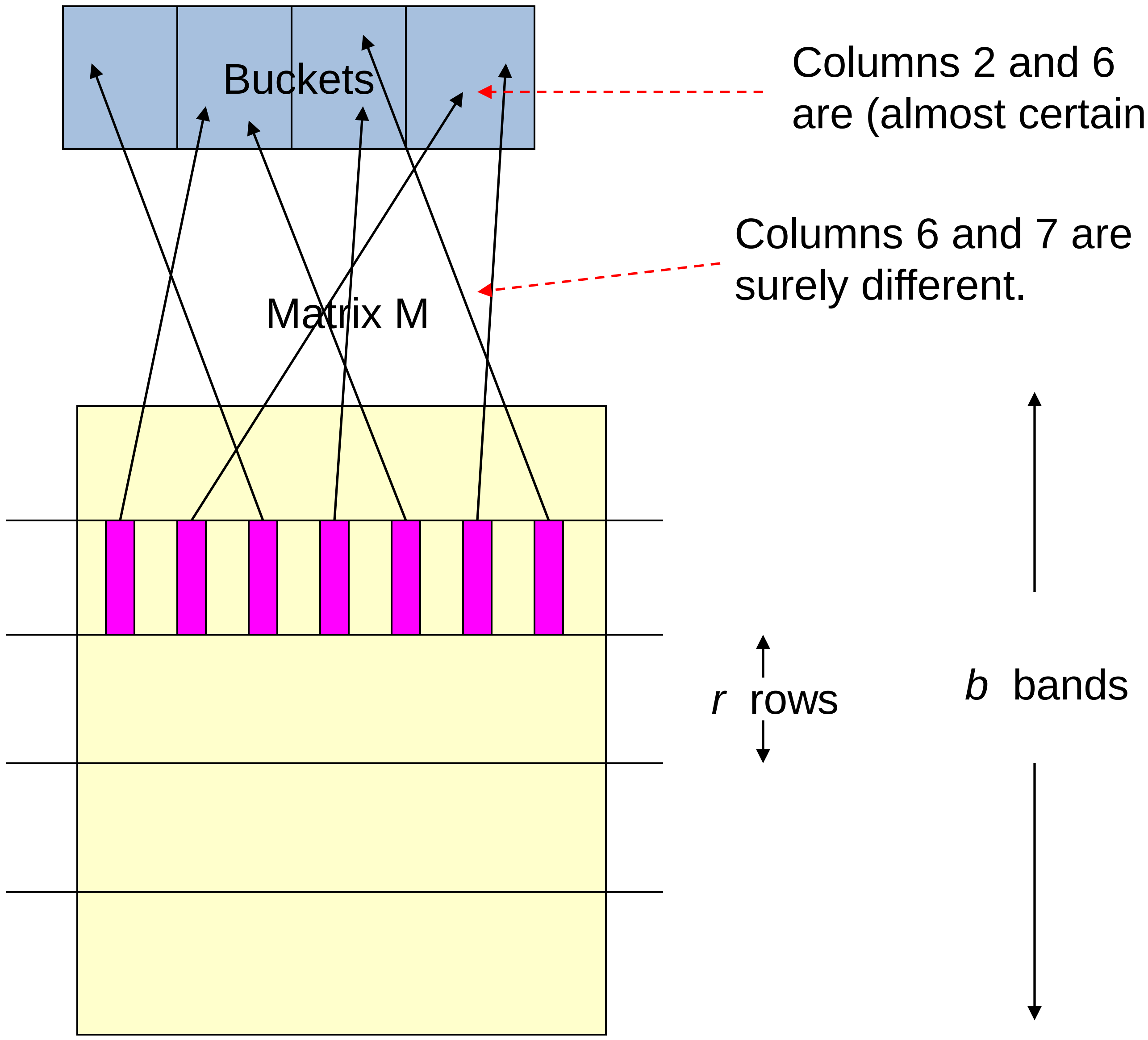
- What we want: a function f(X,Y) that tells whether or not X and Y is a candidate pair: a pair of elements whose similarity must be evaluated.
- A simple idea: X and Y are a candidate pair if they have the same min-hash signature.
  - Easy to test by hashing the signatures.
  - Similar sets are more likely to have the same signature.
  - Likely to produce many false negatives.
- Making it more complex: Perform this process multiple times; candidate pairs should have at least one common signature.
  - Reduce the probability for false negatives.

The signature matirx



Partition into Bands – (2)

- Divide the signature matrix Sig into *b* bands of *r* rows.
  - Each band is a mini-signature with r hash functions.
- For each band, hash the mini-signature to a hash table with *k* buckets.
  - Make *k* as large as possible so that mini-signatures that hash to the same bucket are almost certainly identical.
- Candidate column pairs are those that hash to the same bucket for $\geq$ 1 band.
- Tune *b* and *r* to catch most similar pairs, but few non-similar pairs.


Buckets
Columns 2 and 6
are (almost certainly) identical.
Columns 6 and 7 are
surely different.
Matrix M
r rows
b bands

Suppose S_1, S_2 are 80% Similar

- We want all 80%-similar pairs. Choose 20 bands of 5 integers/band.

- Probability S_1, S_2 identical in one particular band:
  $(0.8)^5 = 0.328$.

- Probability S_1, S_2 are not similar in any of the 20 bands:
  $(1-0.328)^{20} = .00035$.

  - i.e., about 1/3000-th of the 80%-similar column pairs are false negatives.

Suppose S_1, S_2 Only 40% Similar

- Probability S_1, S_2 identical in any one particular band:

  $$(0.4)^5 = 0.01 .$$

- Probability S_1, S_2 identical in at least 1 of 20 bands:

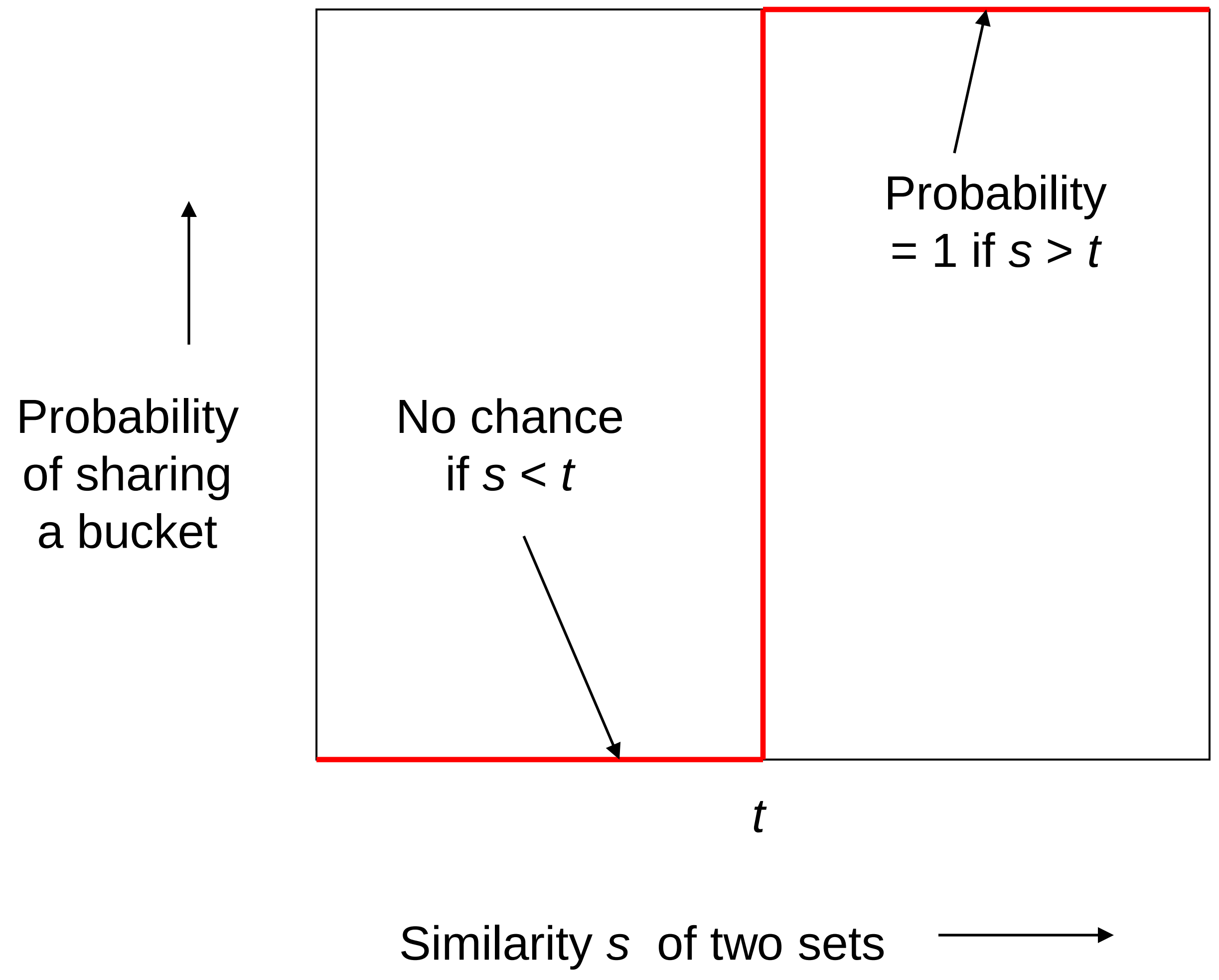
  $$\leq 20 * 0.01 = 0.2 .$$

- But false positives much lower for similarities << 40%.

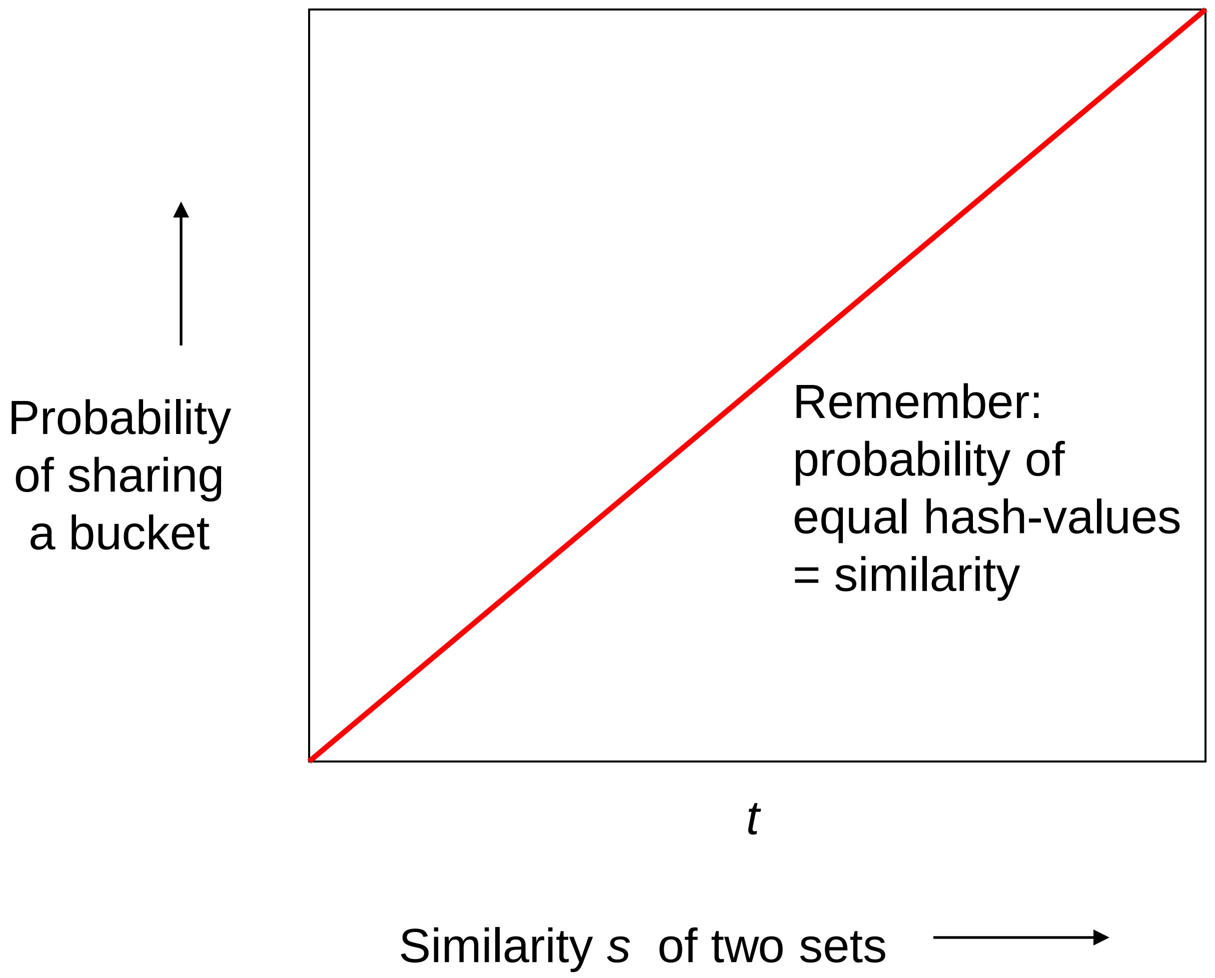
LSH Involves a Tradeoff

- Pick the number of minhashes, the number of bands, and the number of rows per band to balance false positives/negatives.
- Example: if we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up.

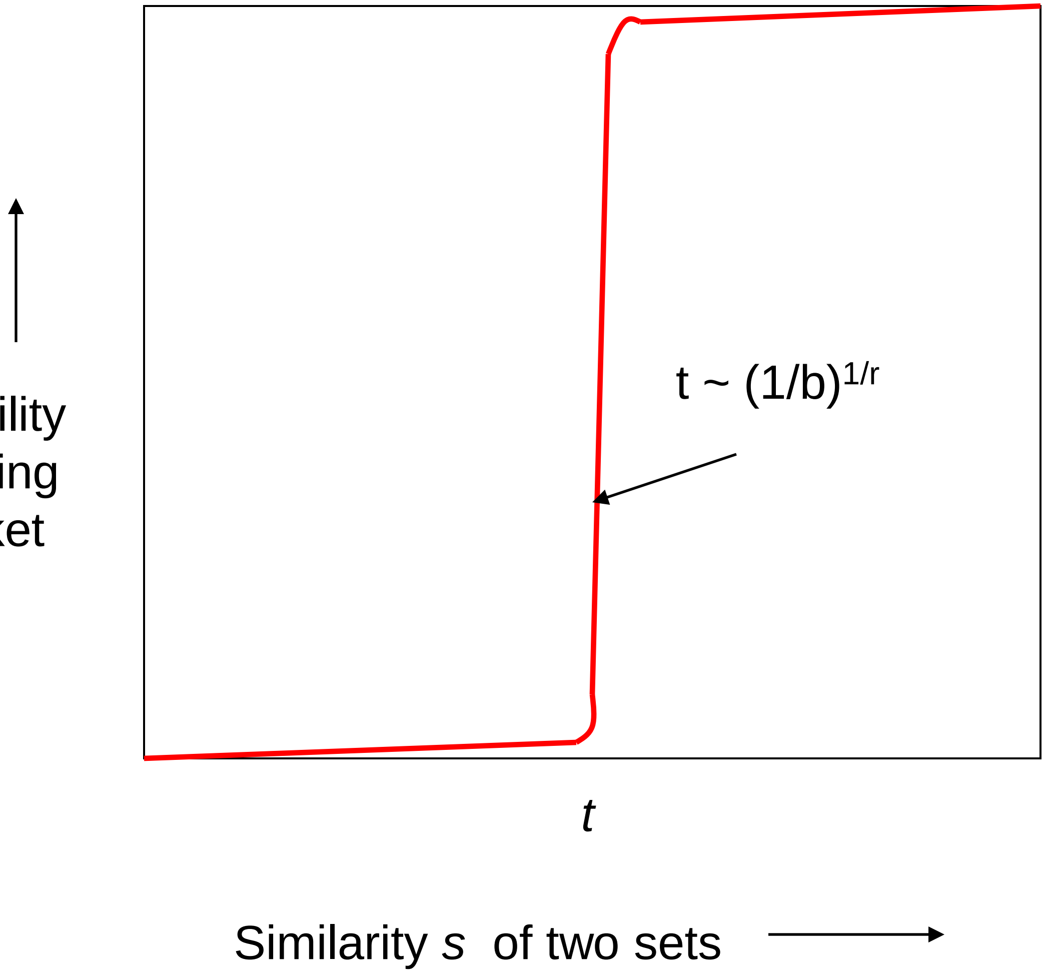
Analysis of LSH – What We Want



What One Band of One Row Gives You



What *b* Bands of *r* Rows Gives You



At least one band identical

No bands identical

$$1 - \quad (1 - \quad s^{\,r} \quad)^{b}$$

Some row of a band unequal

All rows of a band are equal

Example: $b = 20$; $r = 5$

s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

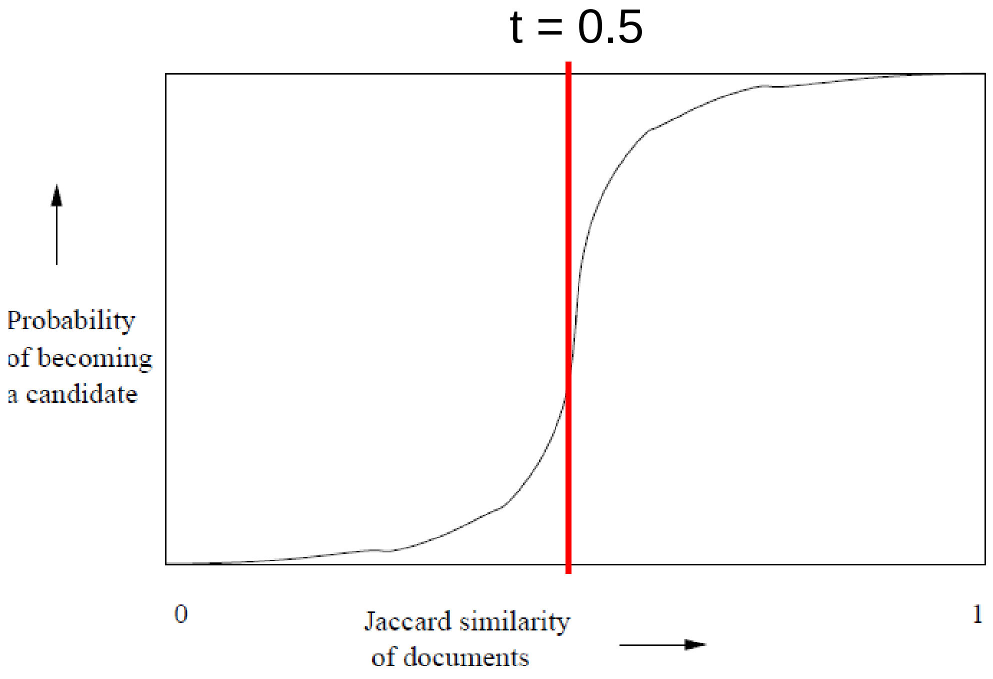


Figure 3.7: The S-curve

Locality-sensitive hashing (LSH)

- Big Picture: Construct hash functions **$h: R^d \rightarrow U$** such that for any pair of points **p,q**, for distance function **D** we have:
  - If **$D(p,q) \leq r$**, then **$Pr[h(p)=h(q)]$** is high
  - If **$D(p,q) \geq cr$**, then **$Pr[h(p)=h(q)]$** is small
- Then, we can find close pairs by hashing

- LSH is a general framework: for a given distance function **D** we need to find the right **h**

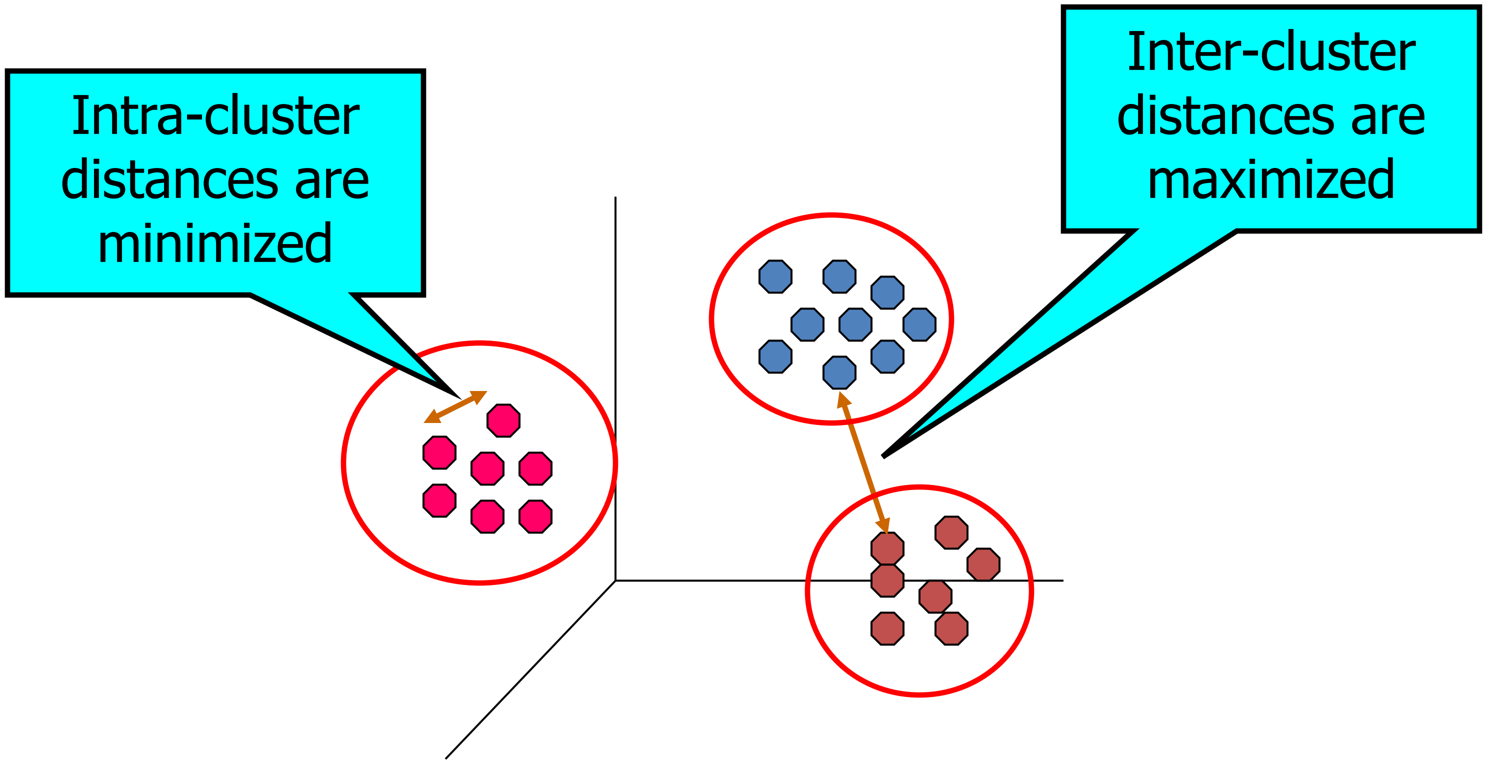
CLUSTERING

Thanks to

Tan, Steinbach, Kumar, “Introduction to Data Mining”

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups

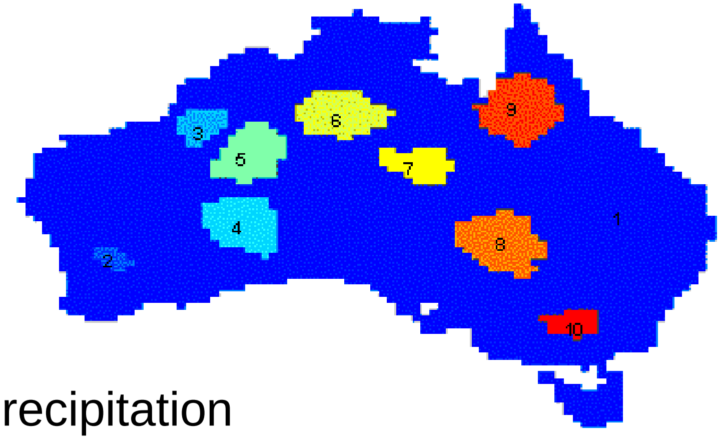

Intra-cluster distances are minimized
Inter-cluster distances are maximized

Applications of Cluster Analysis

- **Understanding**
  - Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations

	Discovered Clusters	*Industry Group*
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP

- **Summarization**
  - Reduce the size of large data sets



Clustering precipitation in Australia

Early applications of cluster analysis

- John Snow, London 1854

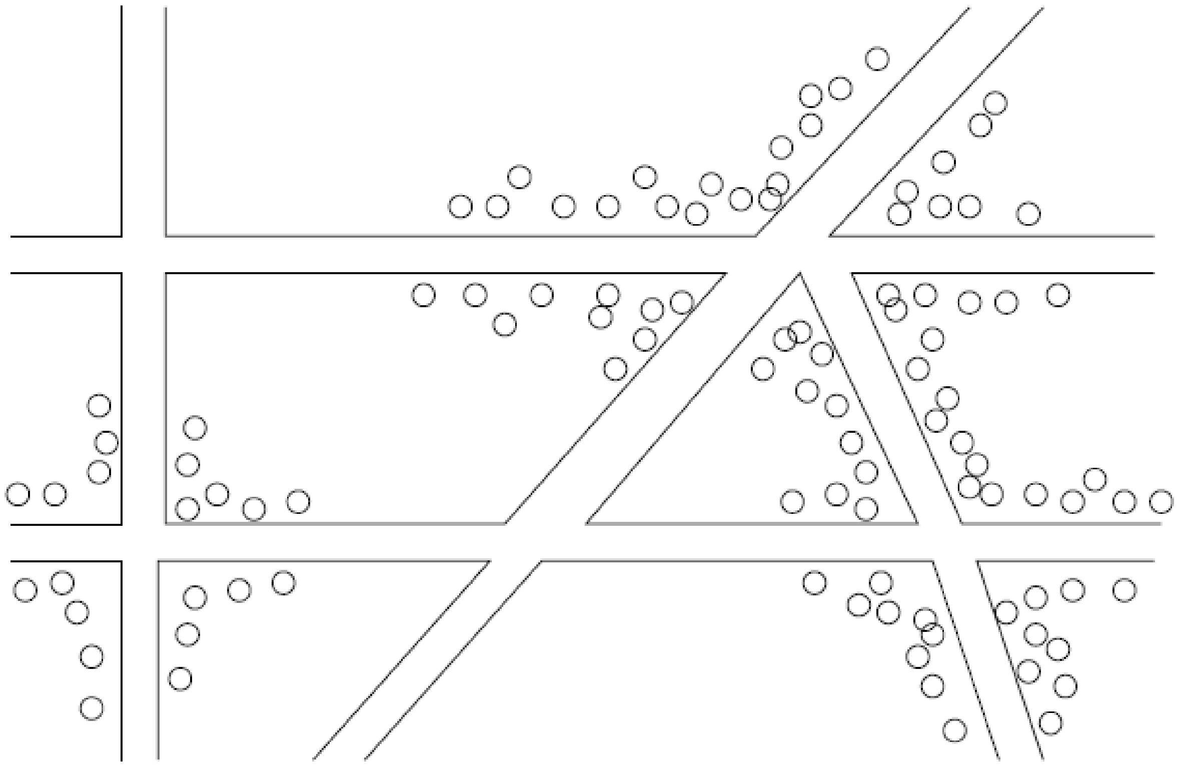
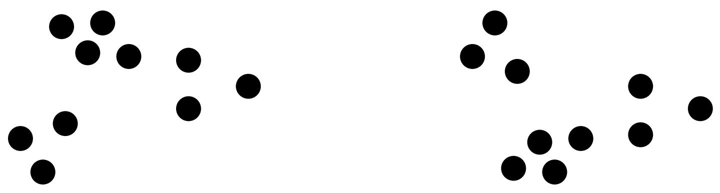
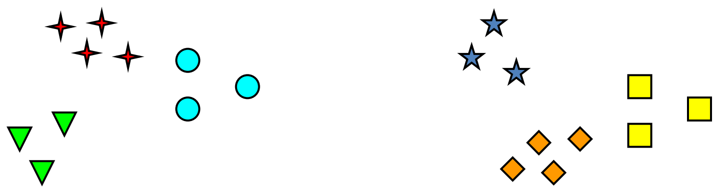
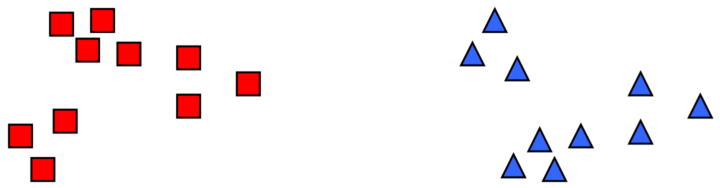


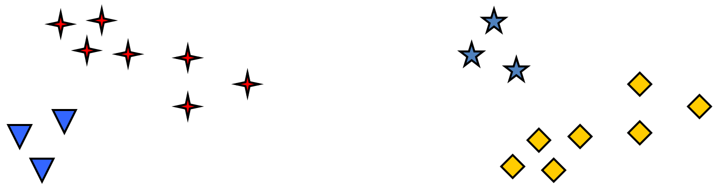
Figure 1.1: Plotting cholera cases on a map of London

Notion of a Cluster can be Ambiguous


How many clusters?


Six Clusters

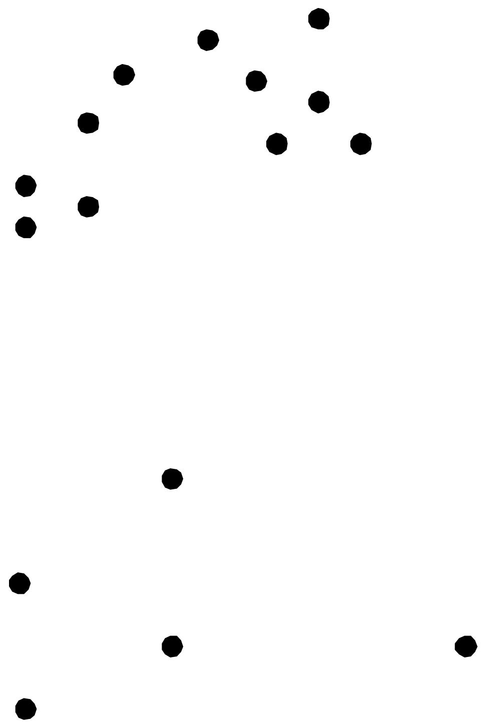

Two Clusters

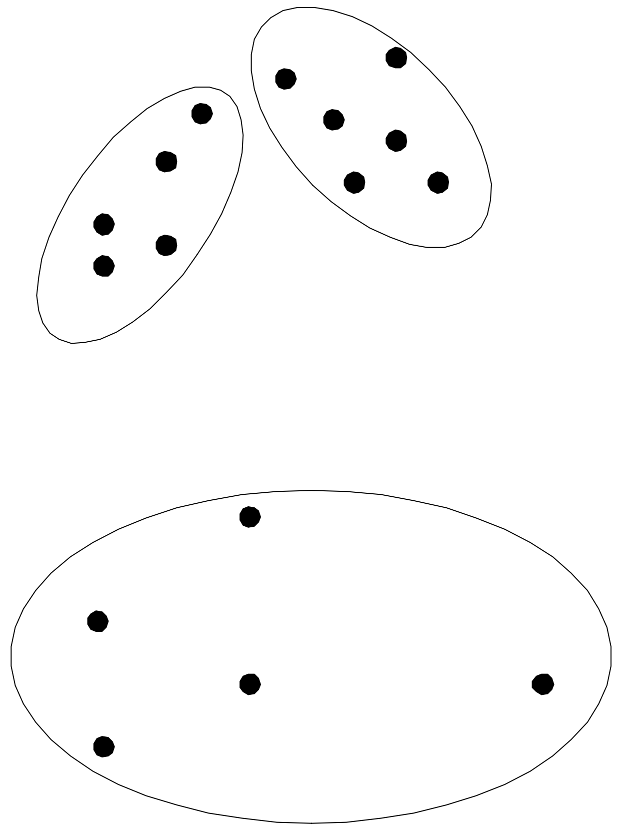

Four Clusters

Types of Clusterings

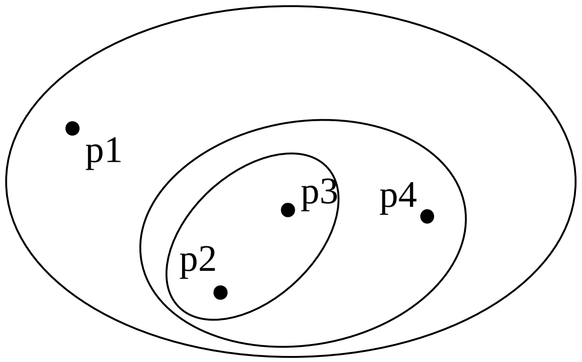
- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

Partitional Clustering

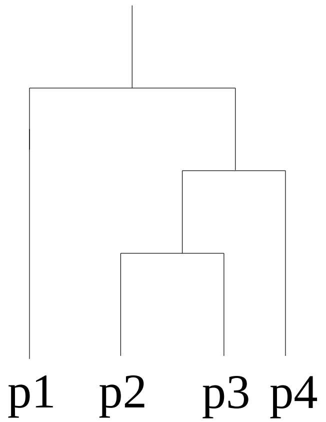

Original Points


A Partitional Clustering

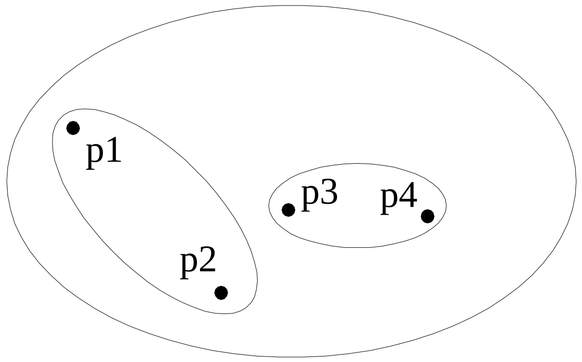
Hierarchical Clustering



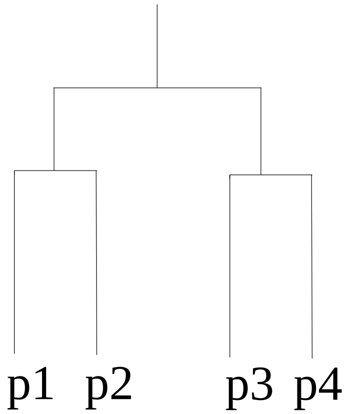
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Other Distinctions Between Sets of Clusters

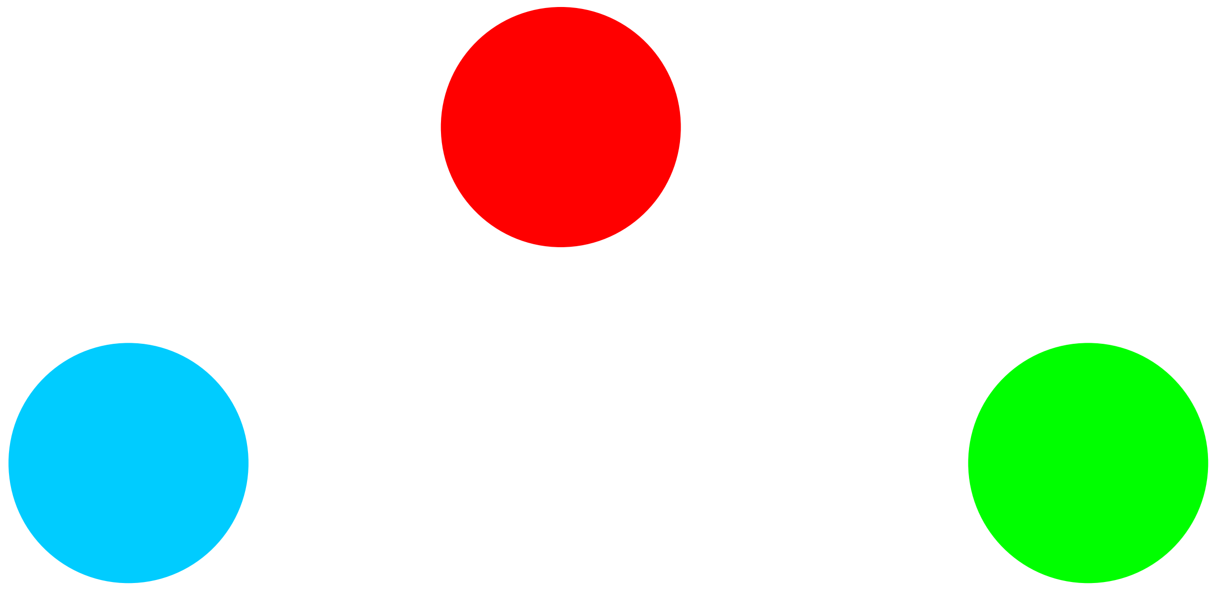
- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or ‘border’ points
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data

Types of Clusters

- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function

Types of Clusters: Well-Separated

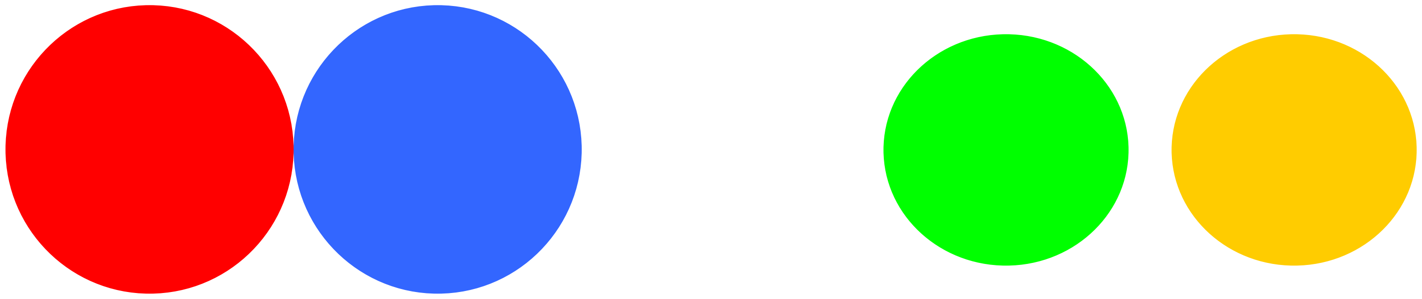
- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters

Types of Clusters: Center-Based

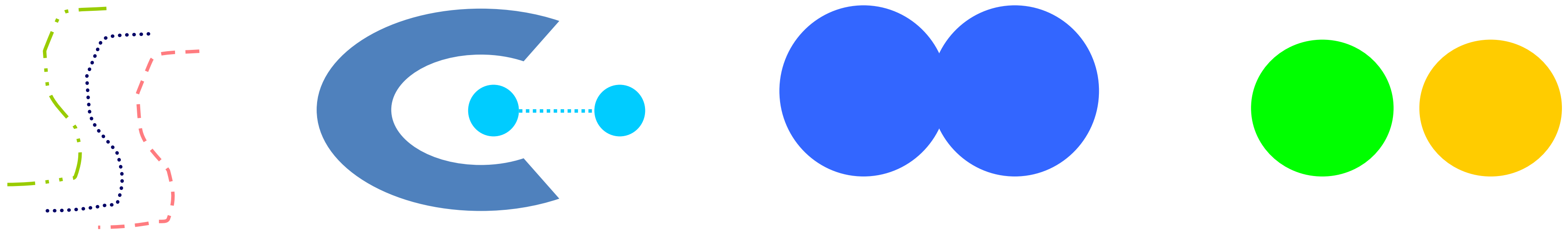
- Center-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster



4 center-based clusters

Types of Clusters: Contiguity-Based

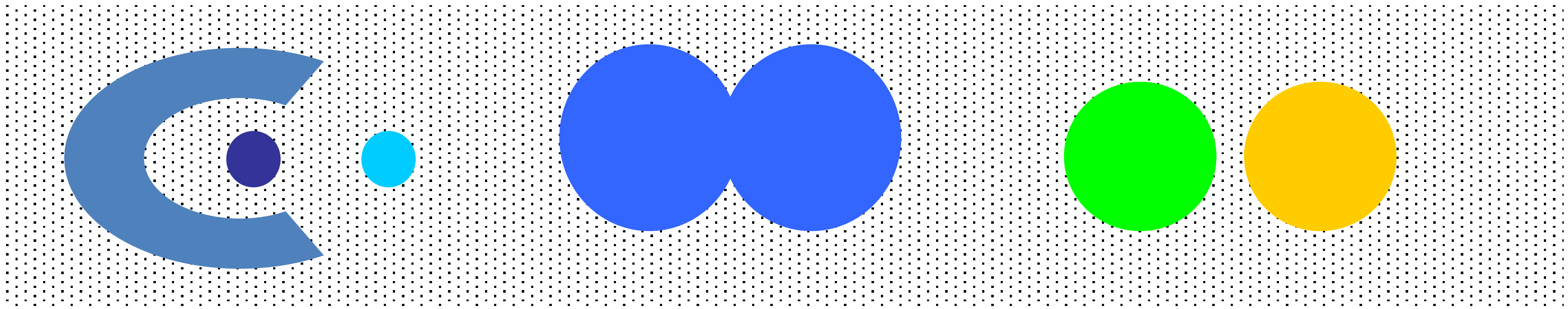
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.



8 contiguous clusters

Types of Clusters: Density-Based

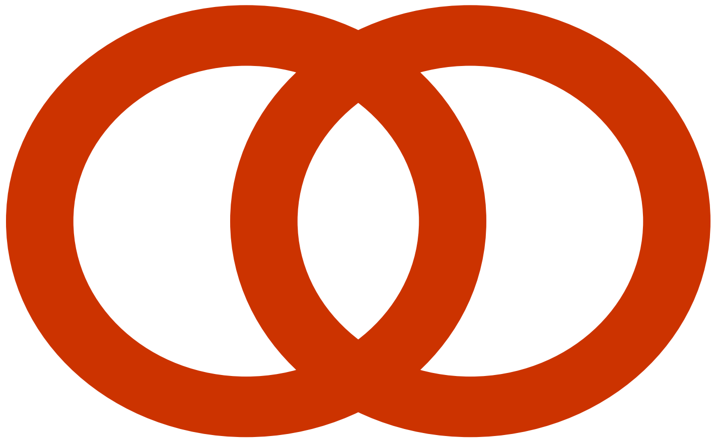
- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.



6 density-based clusters

Types of Clusters: Conceptual Clusters

- Shared Property or Conceptual Clusters
  - Finds clusters that share some common property or represent a particular concept.
  - .



2 Overlapping Circles

Types of Clusters: Objective Function

- Clusters Defined by an Objective Function
  - Finds clusters that minimize or maximize an objective function.
  - Enumerate all possible ways of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters by using the given objective function. (NP Hard)
  - Can have global or local objectives.
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives
  - A variation of the global objective function approach is to fit the data to a parameterized model.
    - Parameters for the model are determined from the data.
    - Mixture models assume that the data is a ‘mixture' of a number of statistical distributions.

Clustering Algorithms

- K-means and its variants
- Hierarchical clustering

K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The objective is to minimize the sum of distances of the points to their respective centroid

K-means Clustering

- Most common definition is with euclidean distance, minimizing the Sum of Squares Error (SSE) function
  - Sometimes K-means is defined like that

- **Problem:** Given a set X of n points in a d-dimensional space and an integer K group the points into K clusters $\{C_1, C_2, \dots, C_k\}$ such that

$$Cost(C) = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(x - c_i)$$

Sum of Squares Error (SSE)

is minimized, where c_i is the mean of the points in cluster C_i

Algorithmic properties of the k-means problem

- NP-hard if the dimensionality of the data is at least 2 (**d>=2**)
- Finding the best solution in polynomial time is infeasible
- For **d=1** the problem is solvable in polynomial time (how?)
- A simple iterative algorithm works quite well in practice

K-means Algorithm

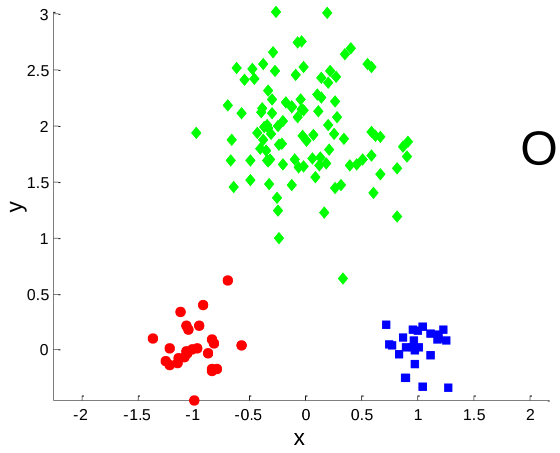
- Also known as Lloyd’s algorithm.
- K-means is sometimes synonymous with this algorithm

```
Select $K$ points as the initial centroids.
repeat
    Form $K$ clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don’t change
```

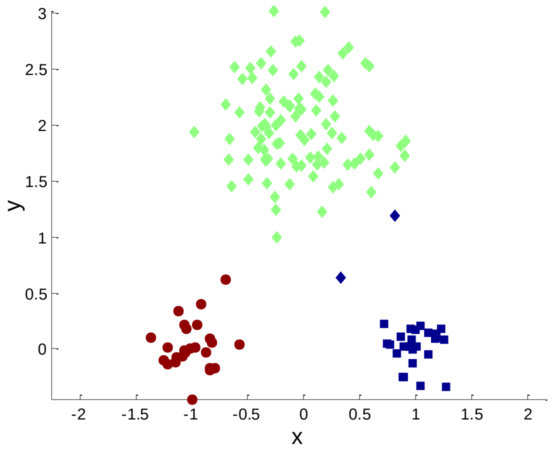
K-means Algorithm – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- The centroid depends on the distance function
  - The mean of the points in the cluster for SSE, the median for Manhattan distance.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes

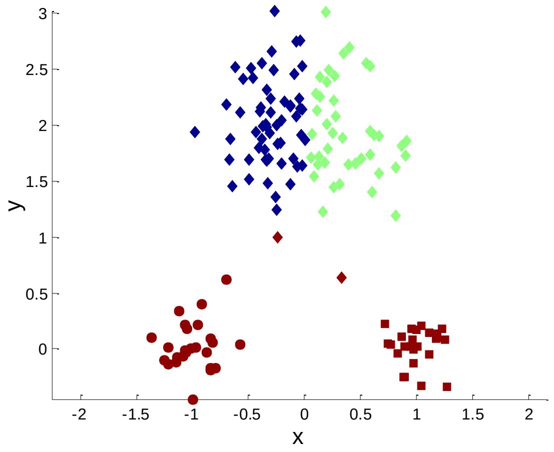
Two different K-means Clusterings



Original Points

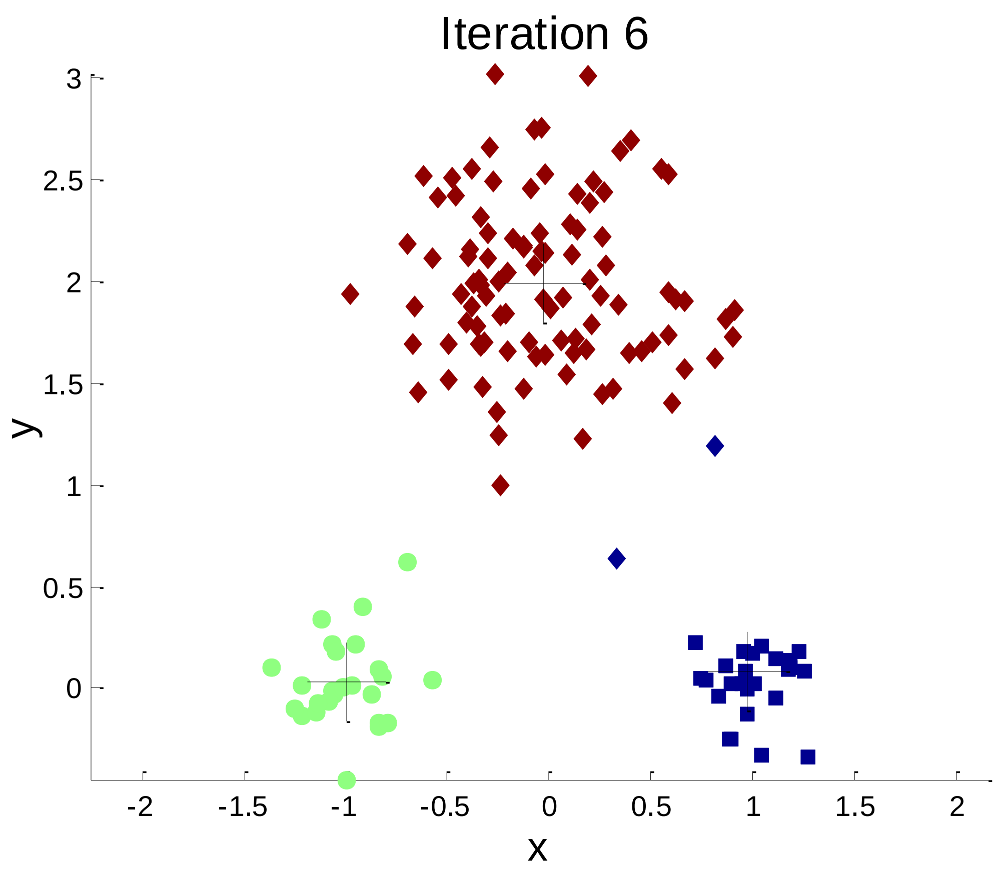


Optimal Clustering

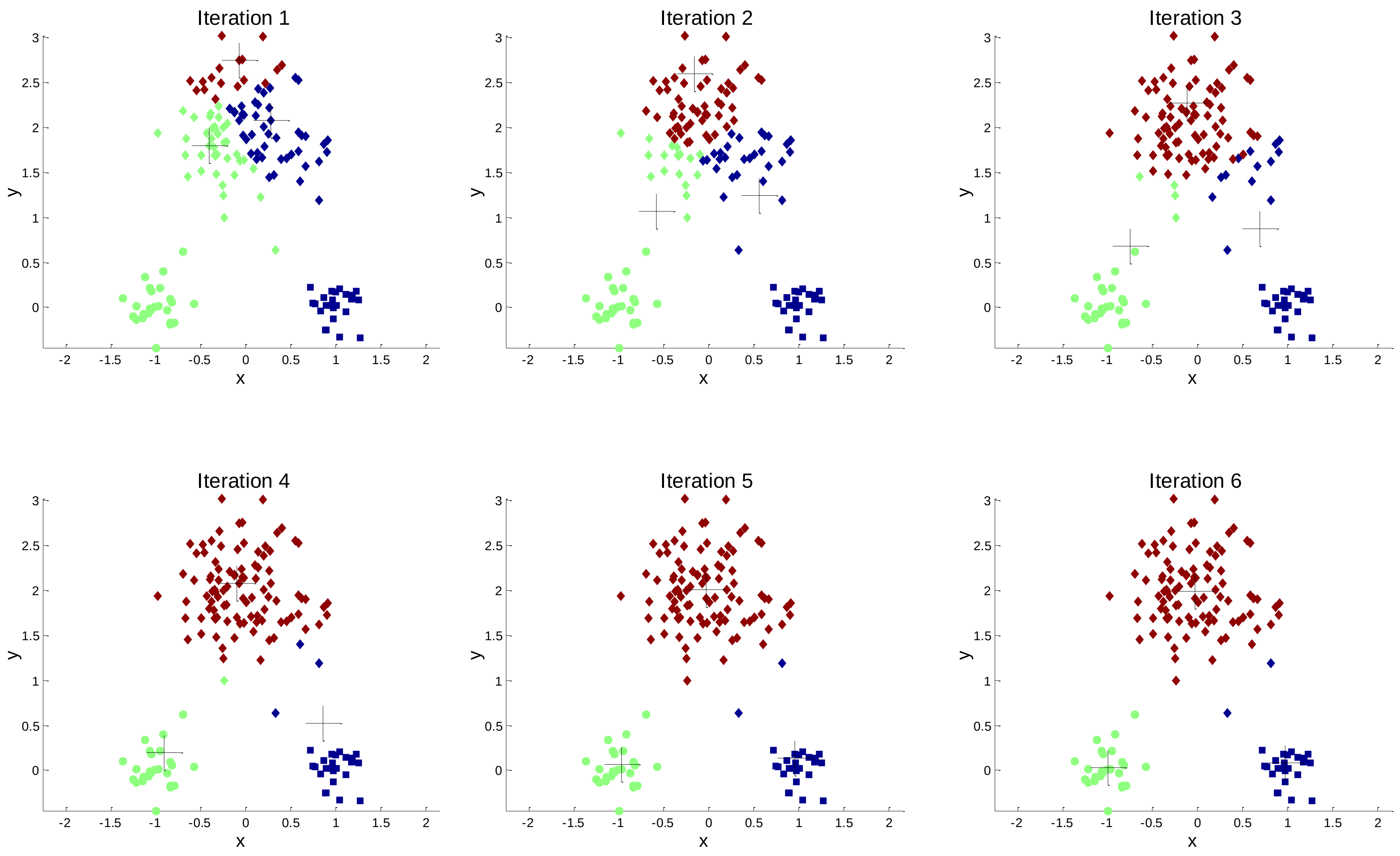


Sub-optimal Clustering

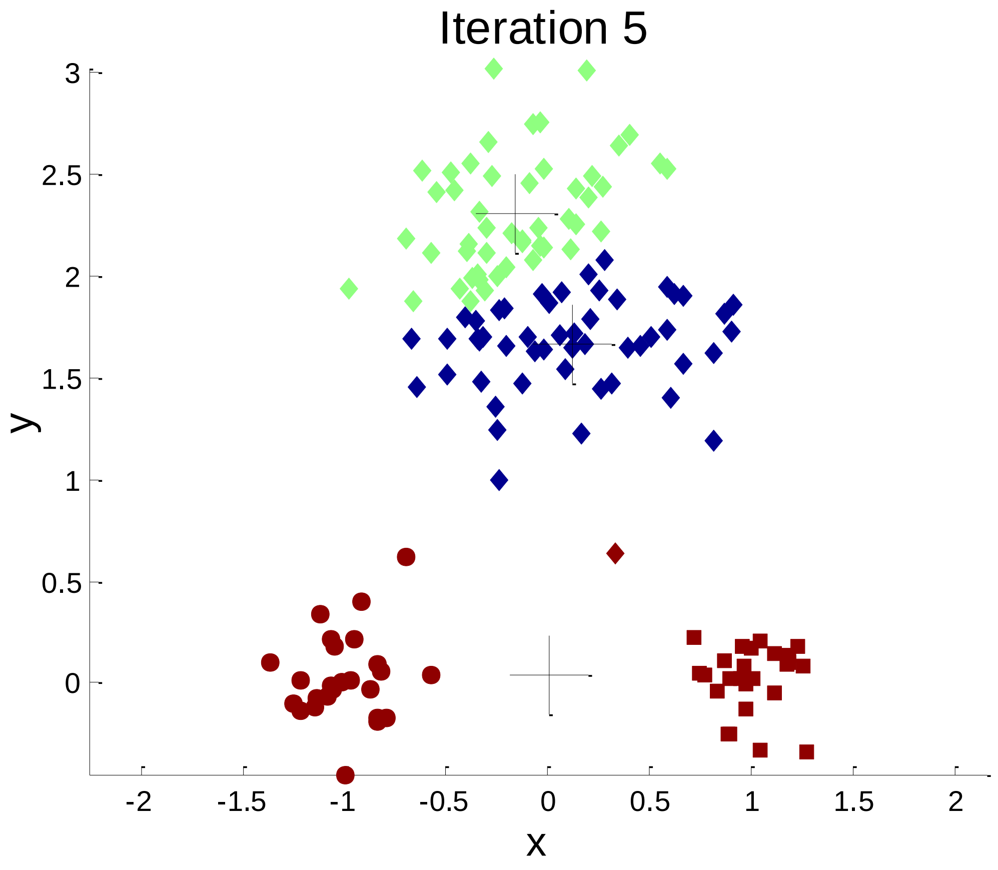
Importance of Choosing Initial Centroids



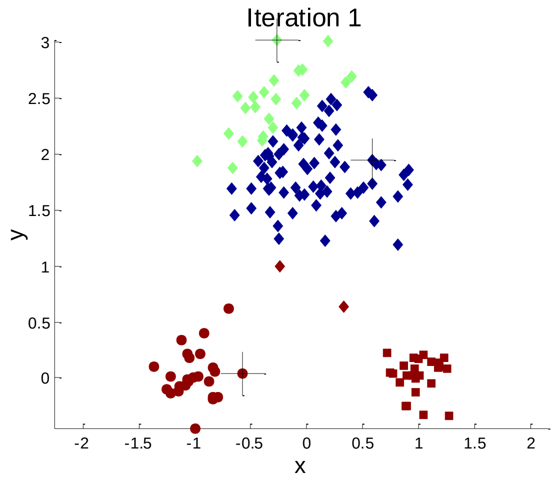
Importance of Choosing Initial Centroids

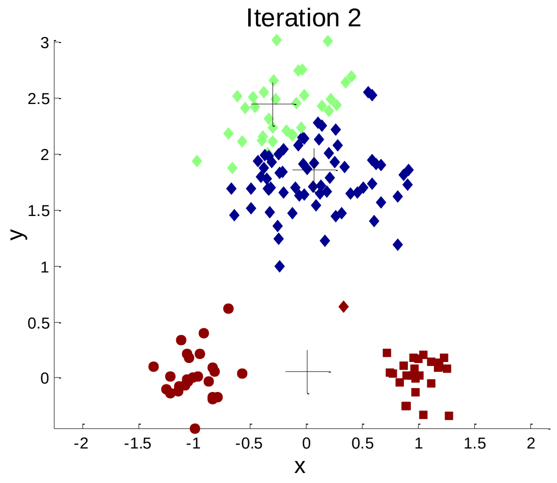


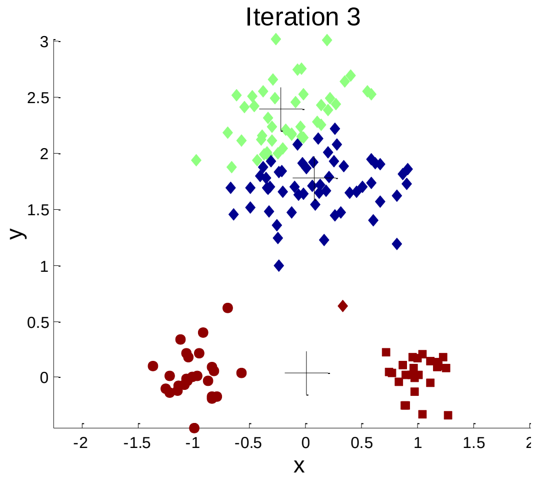
Importance of Choosing Initial Centroids

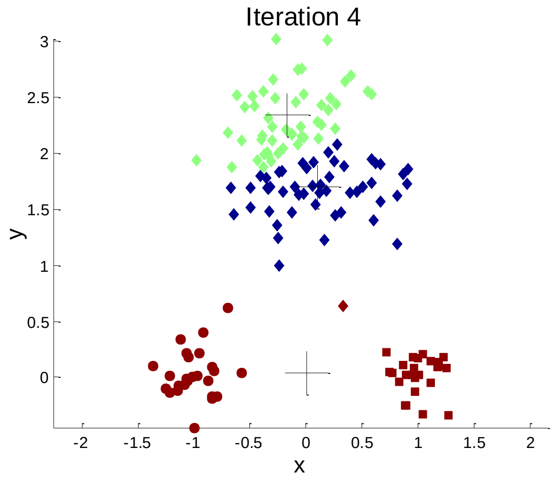


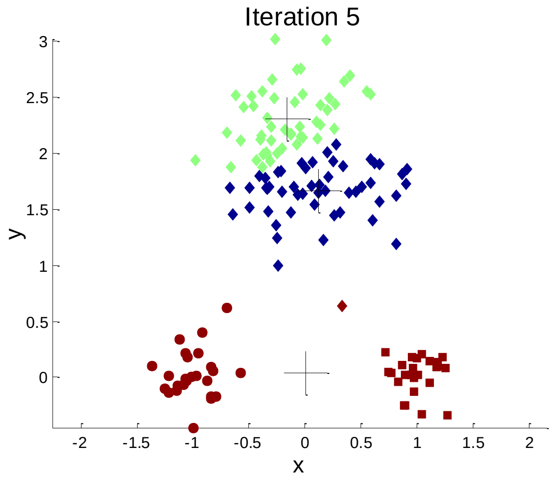
Importance of Choosing Initial Centroids …











Dealing with Initialization

- Do multiple runs and select the clustering with the smallest error

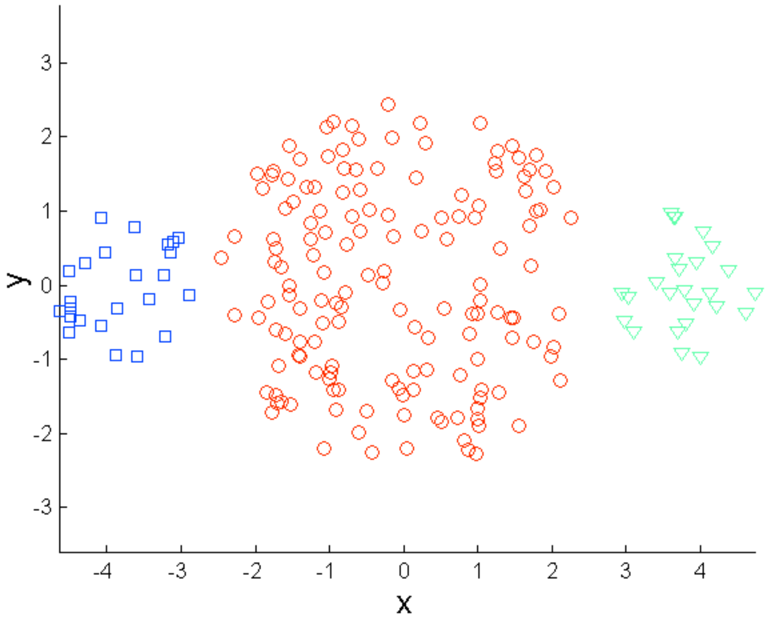
- Select original set of points by methods other than random . E.g., pick the most distant (from each other) points as cluster centers (K-means++ algorithm)

Limitations of K-means

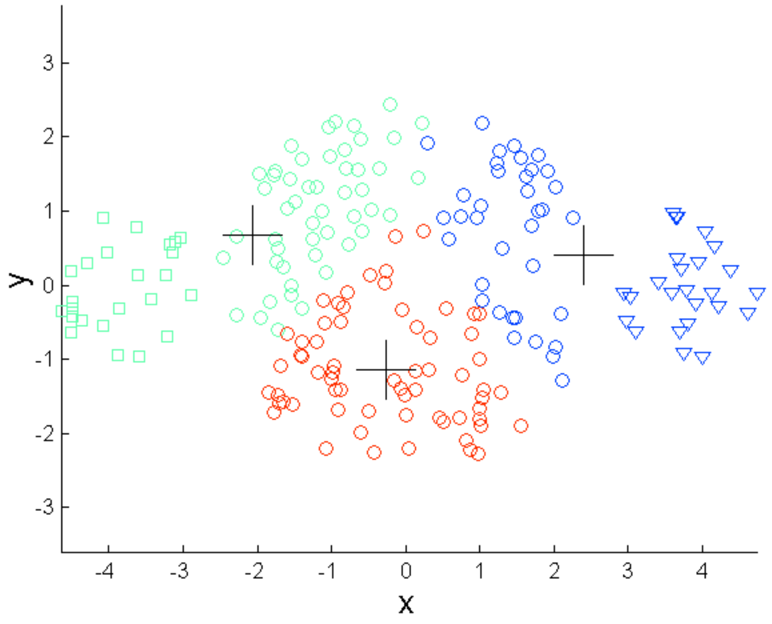
- K-means has problems when clusters are of different
  - Sizes
  - Densities
  - Non-globular shapes

- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

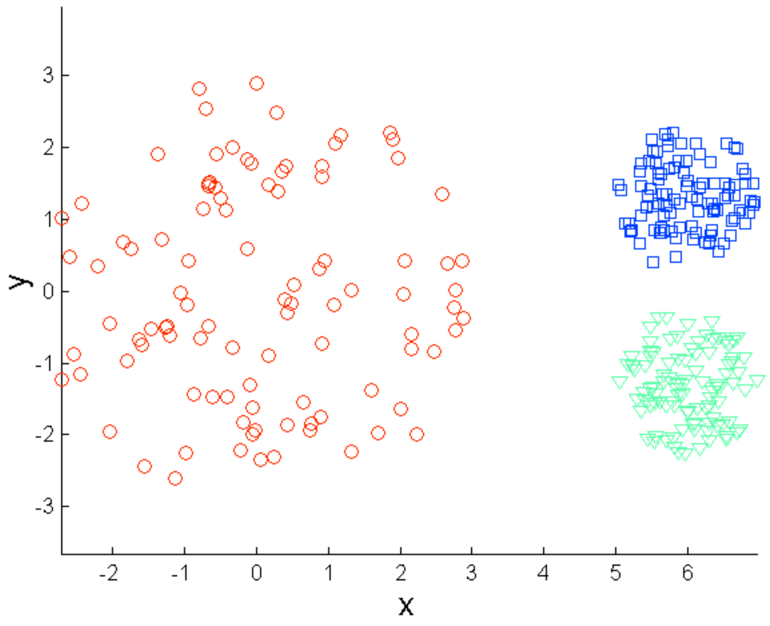


Original Points

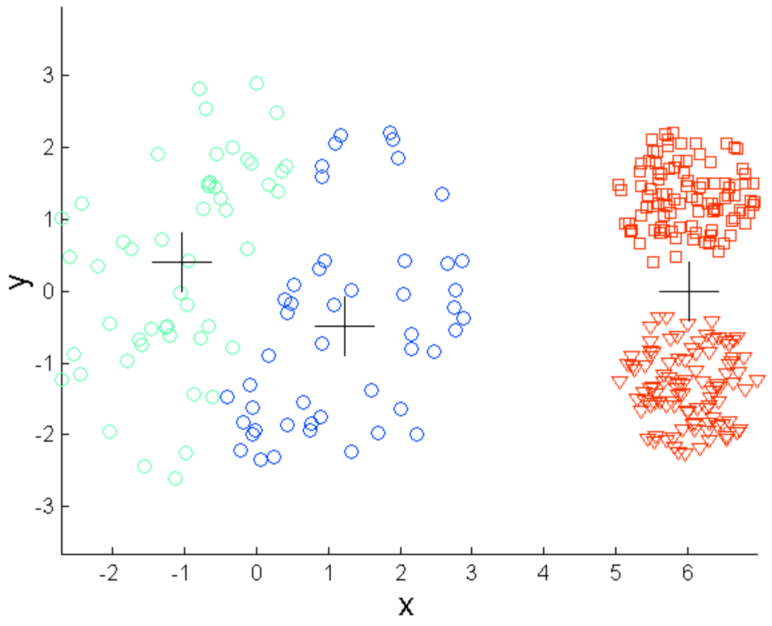


K-means (3 Clusters)

Limitations of K-means: Differing Density

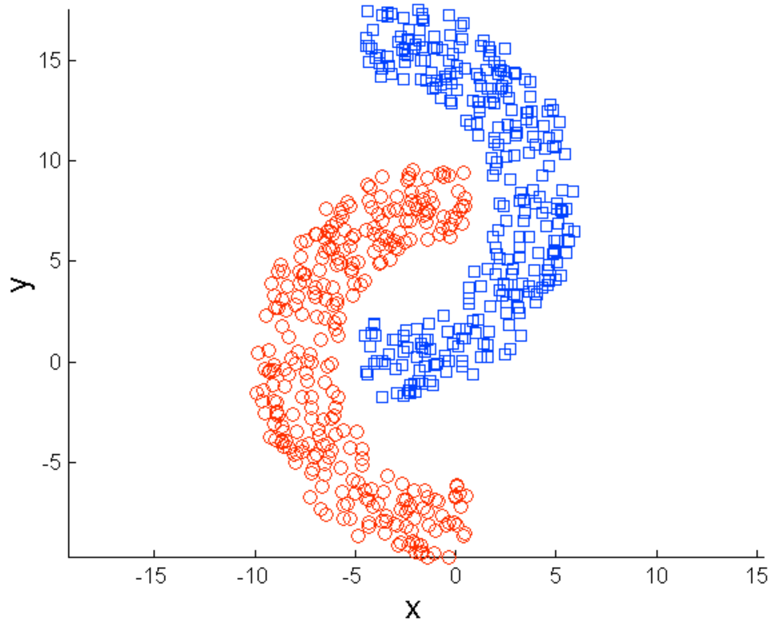


Original Points

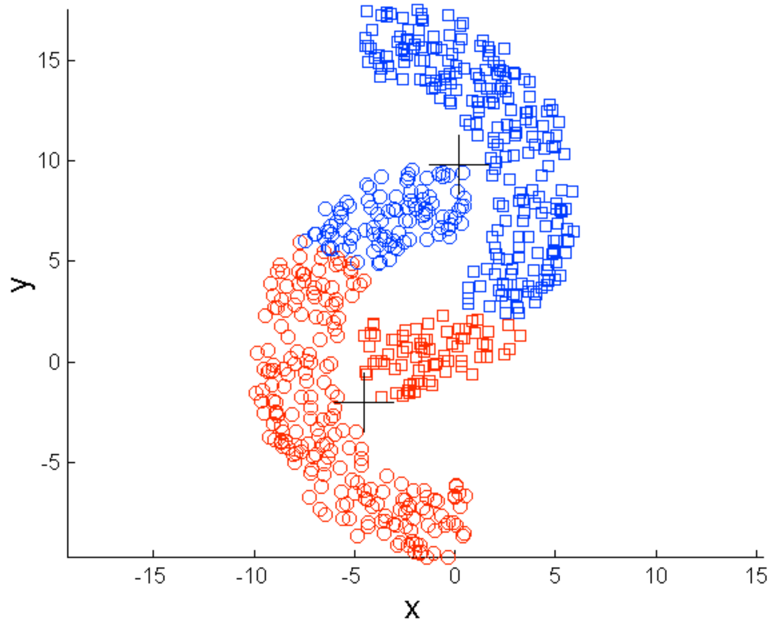


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

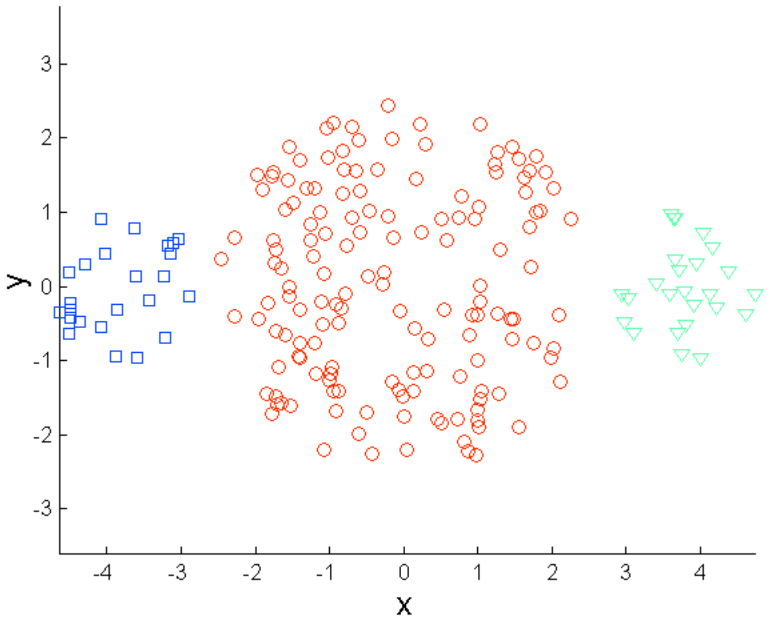


Original Points

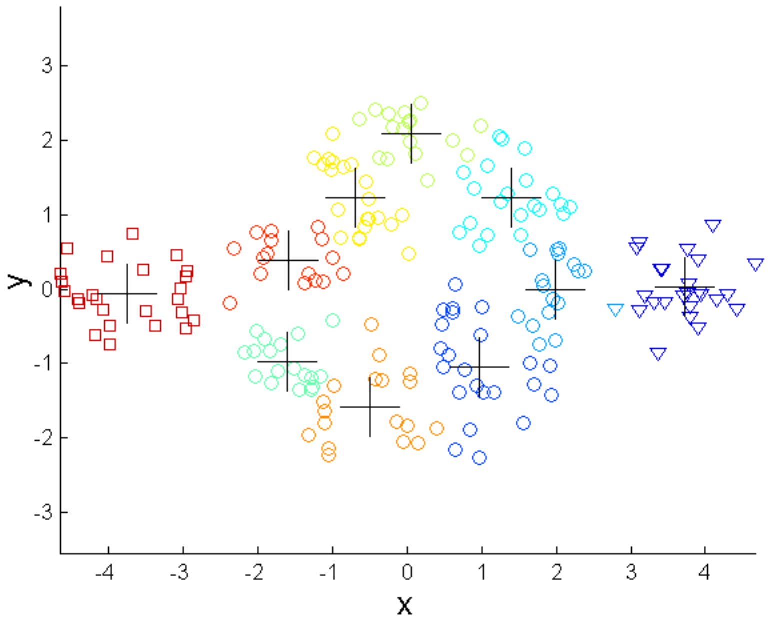


K-means (2 Clusters)

Overcoming K-means Limitations



Original Points

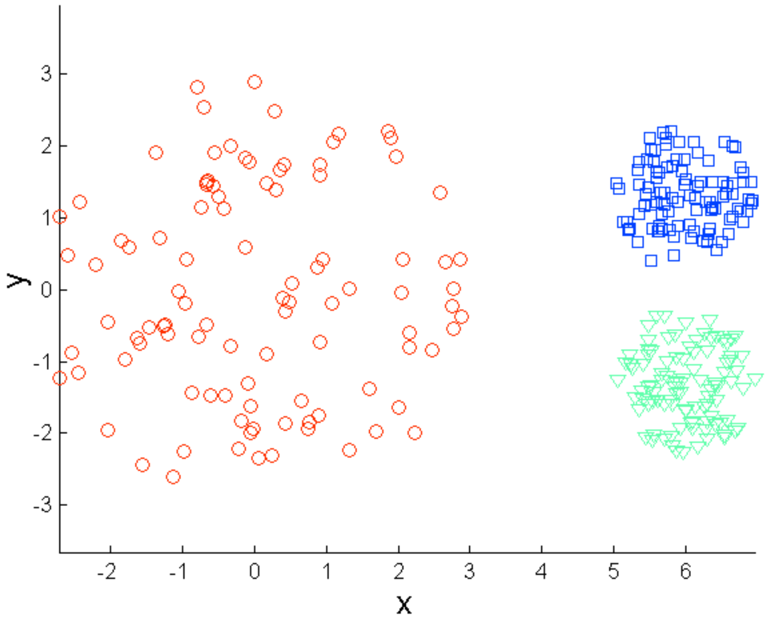


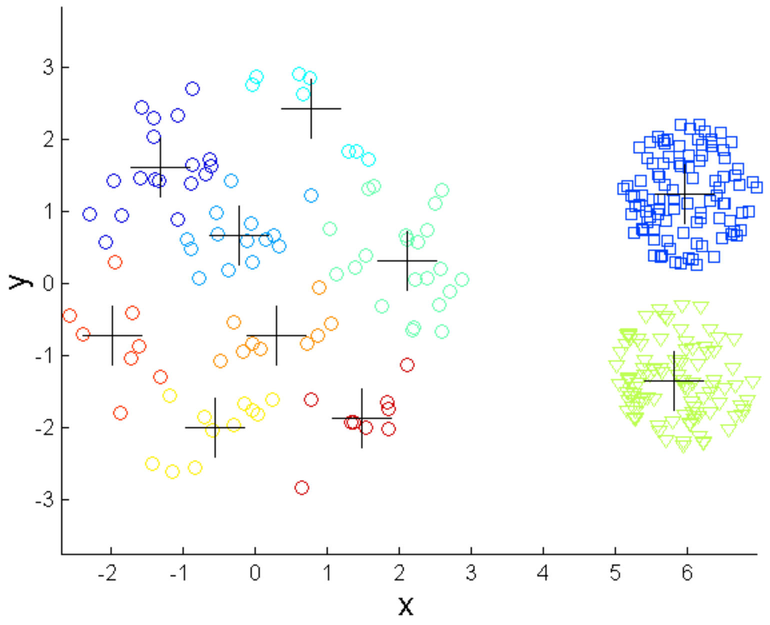
K-means Clusters

One solution is to use many clusters.

Find parts of clusters, but need to put together.

Overcoming K-means Limitations

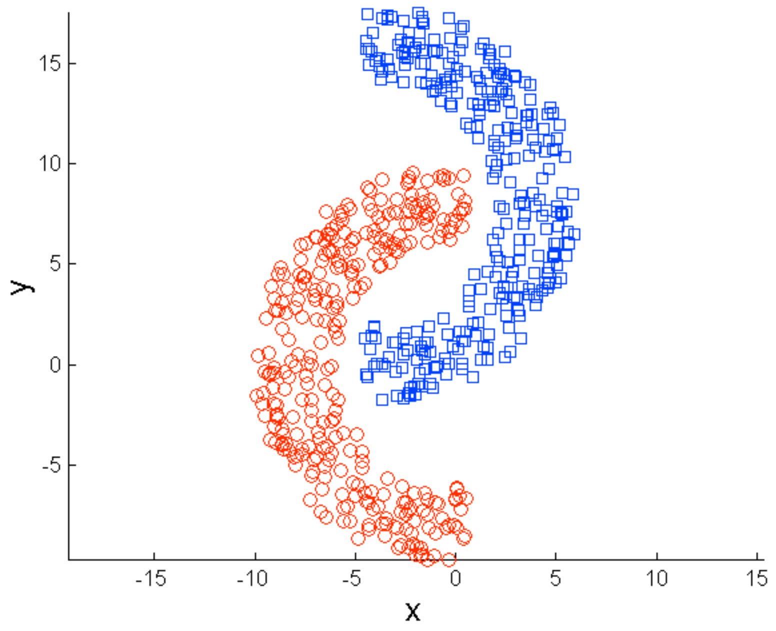


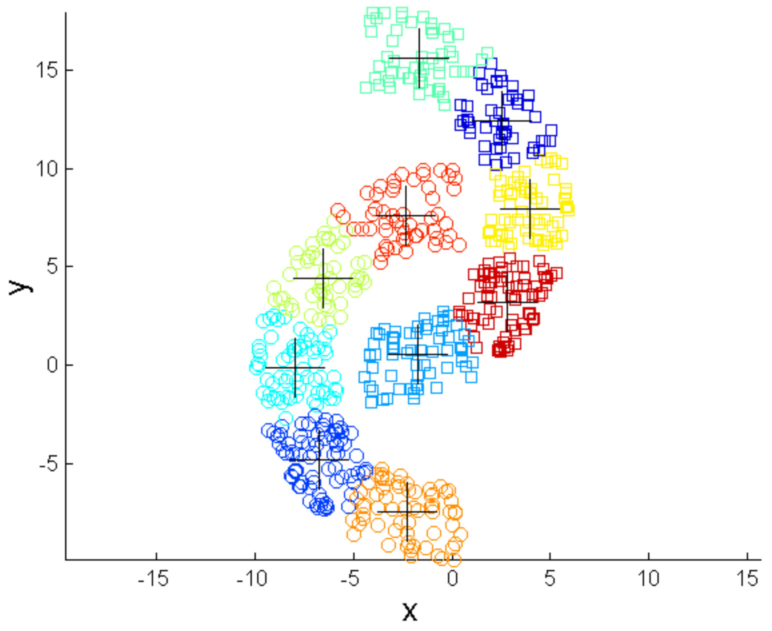


Original Points

K-means Clusters

Overcoming K-means Limitations





Original Points

K-means Clusters

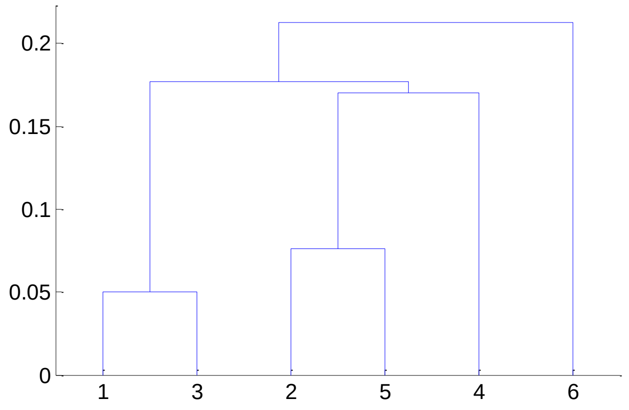
Variations

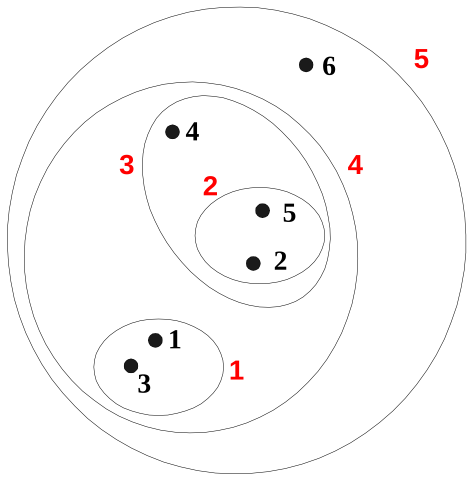
- K-medoids: Similar problem definition as in K-means, but the centroid of the cluster is defined to be one of the points in the cluster (the medoid).

- K-centers: Similar problem definition as in K-means, but the goal now is to minimize the maximum diameter of the clusters (diameter of a cluster is maximum distance between any two points in the cluster).

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level

- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

Hierarchical Clustering

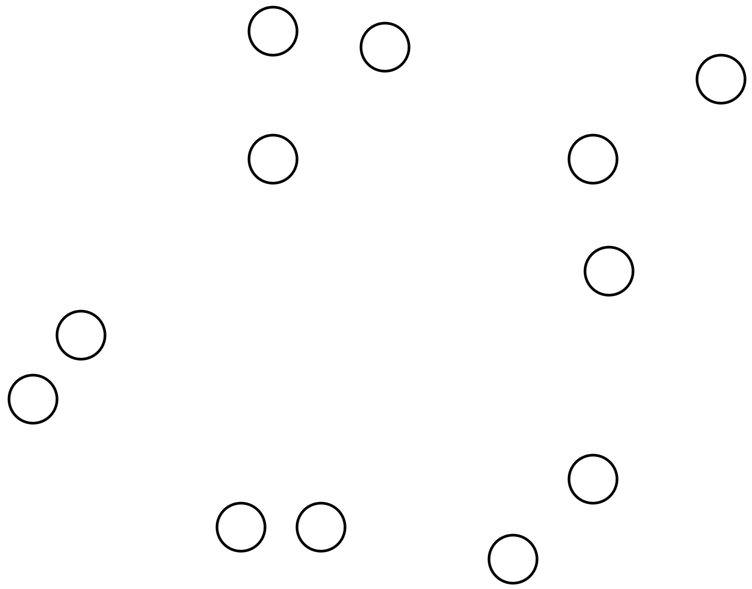
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

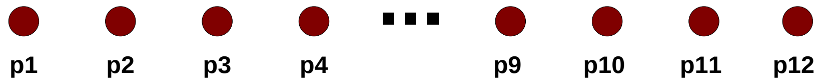
Starting Situation

- Start with clusters of individual points and a proximity matrix



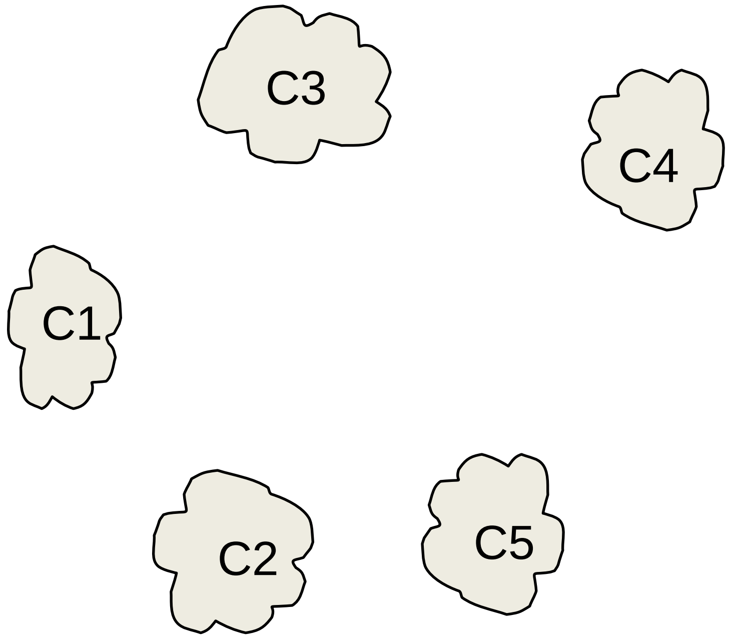
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



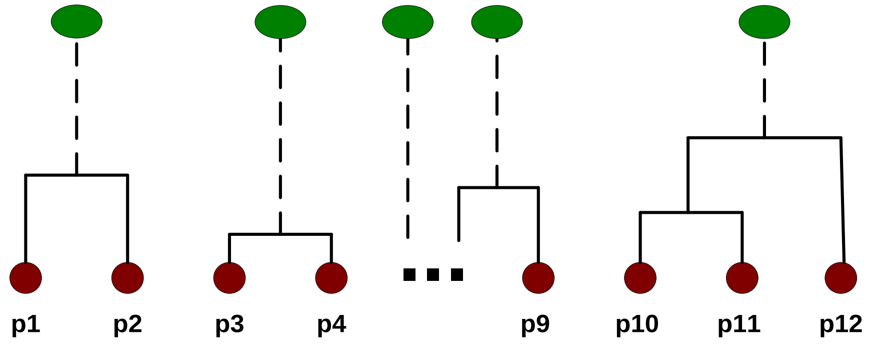
Intermediate Situation

- After some merging steps, we have some clusters



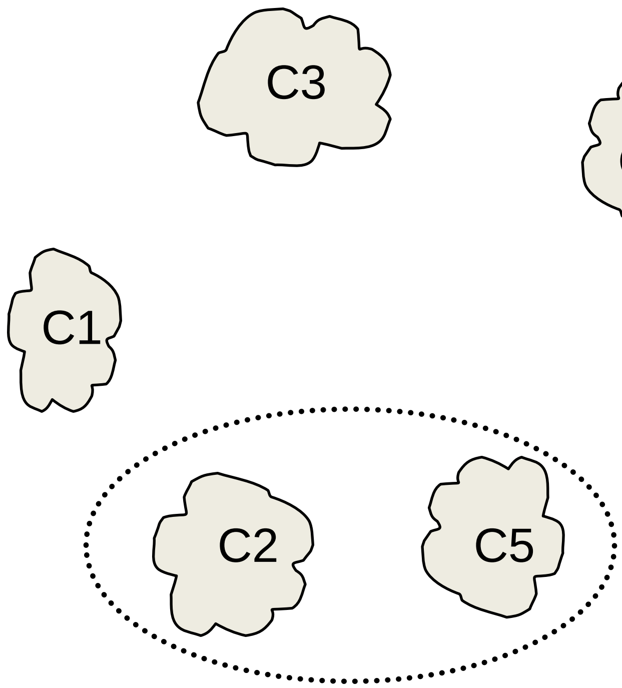
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

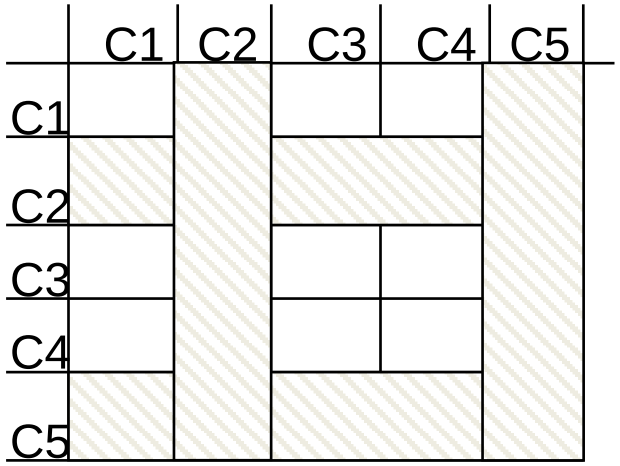
Proximity Matrix



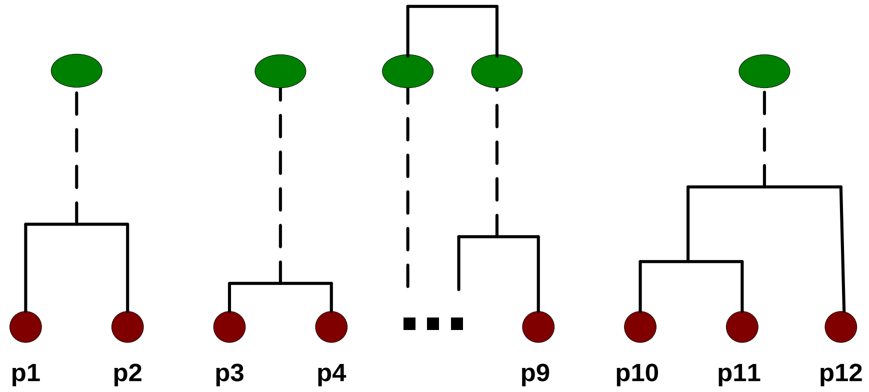
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



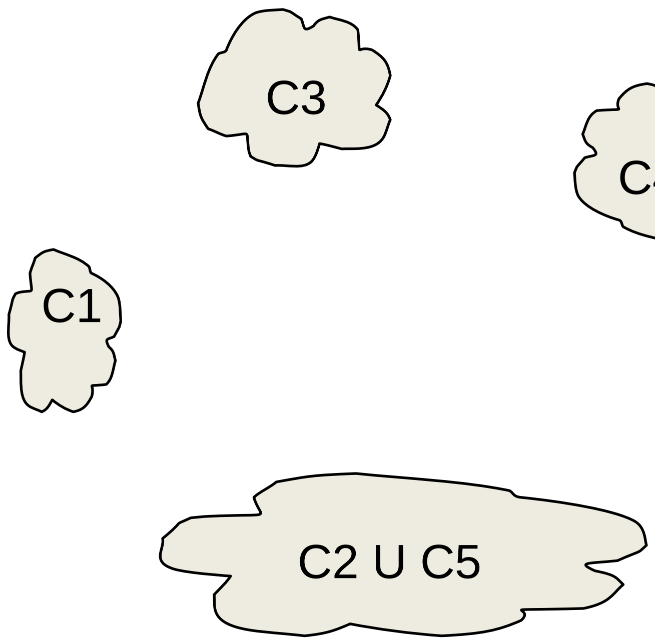


Proximity Matrix



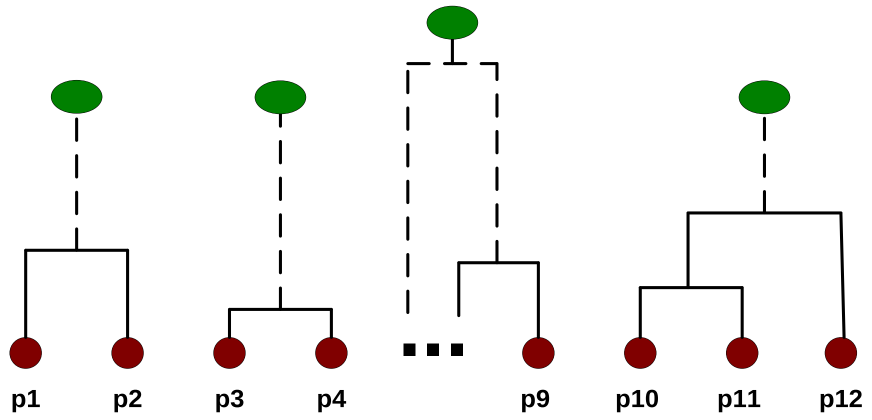
After Merging

- The question is “How do we update the proximity matrix?”

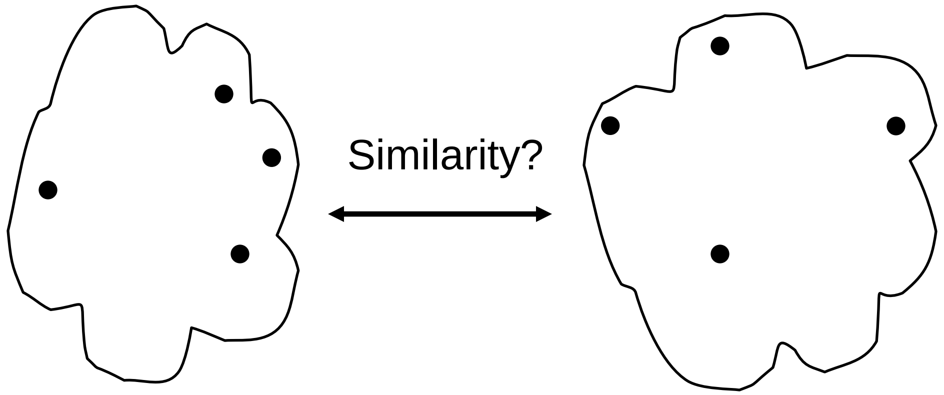


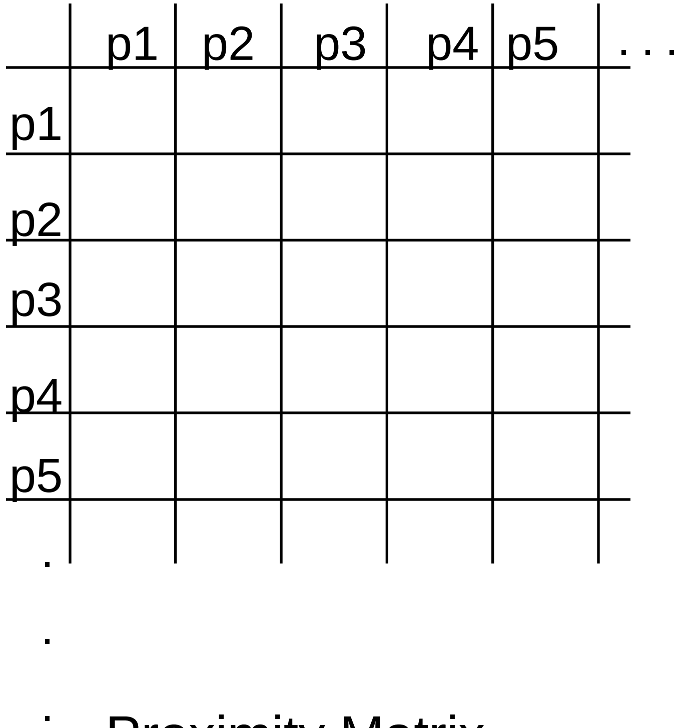
	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

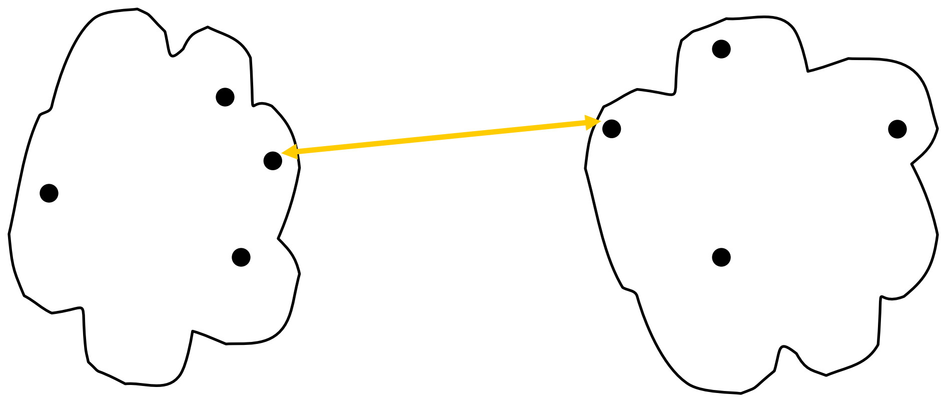




Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

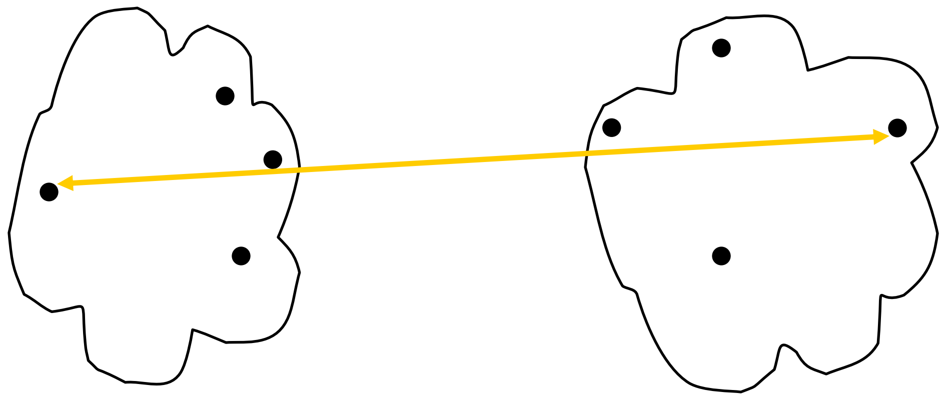


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

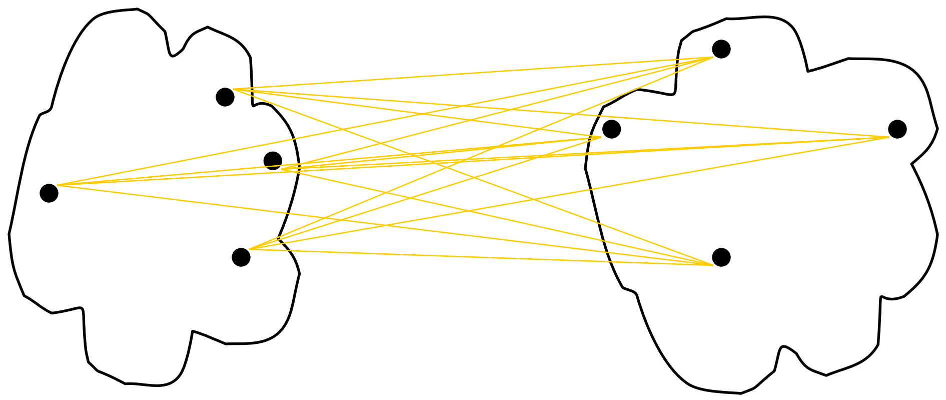


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  – Ward’s Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

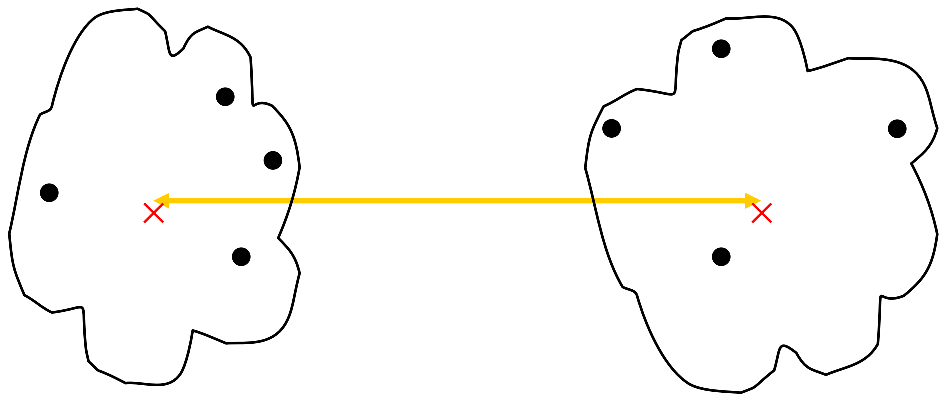


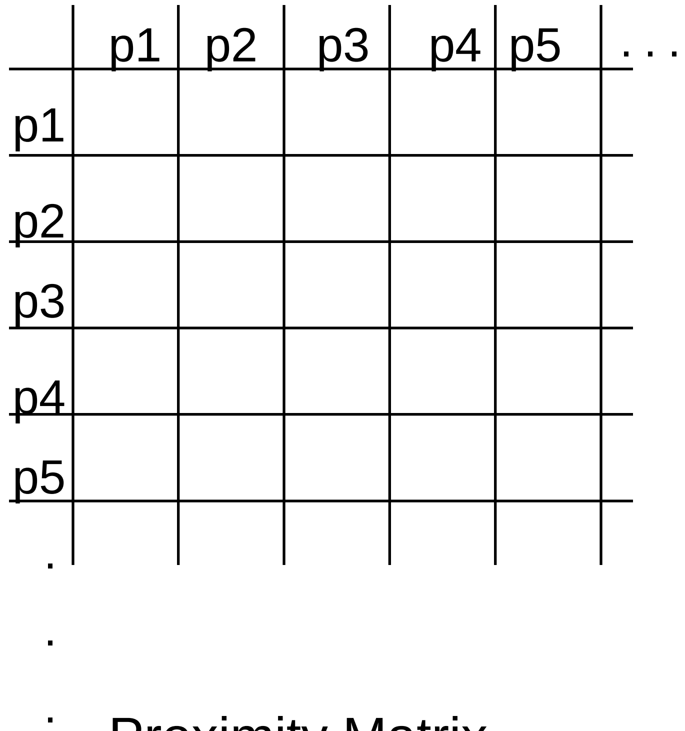
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward’s Method uses squared error

How to Define Inter-Cluster Similarity





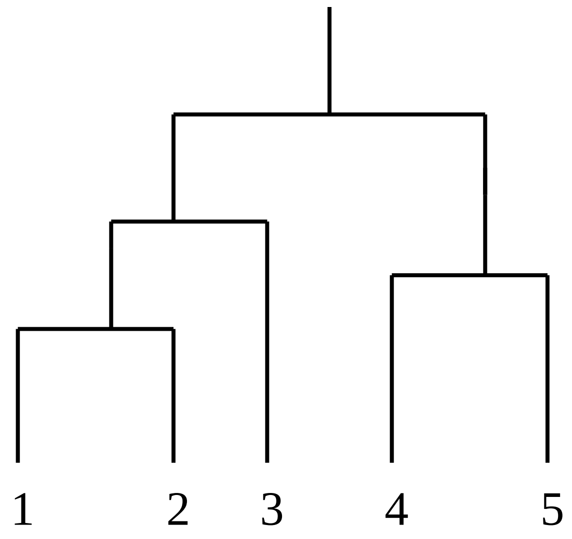
Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

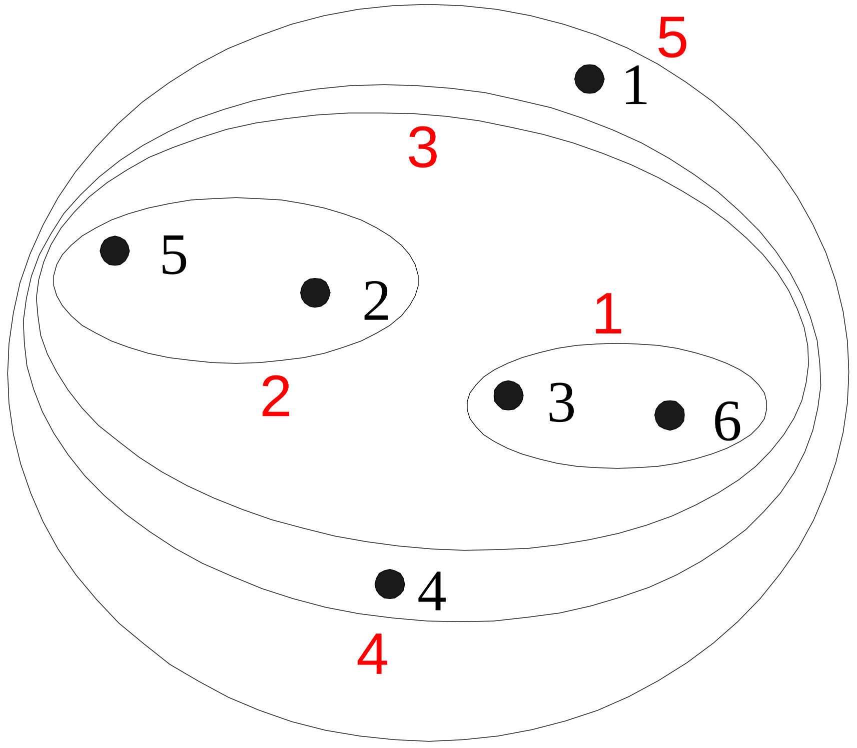
Cluster Similarity: MIN or Single Link

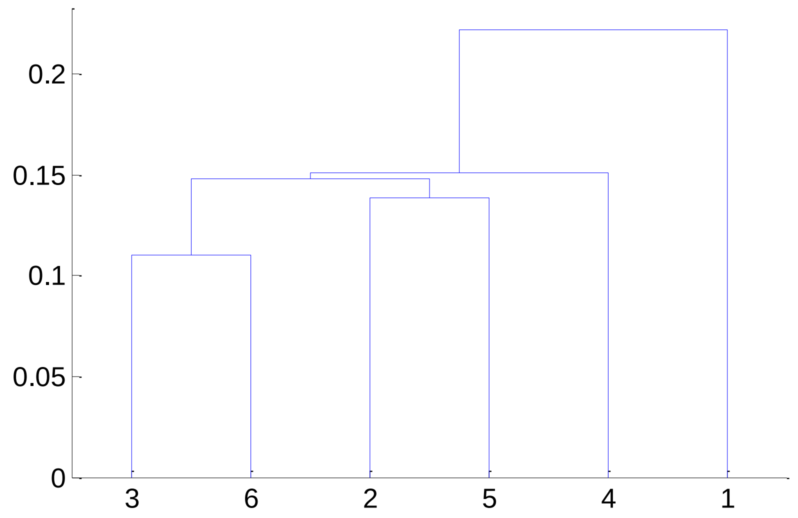
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MIN

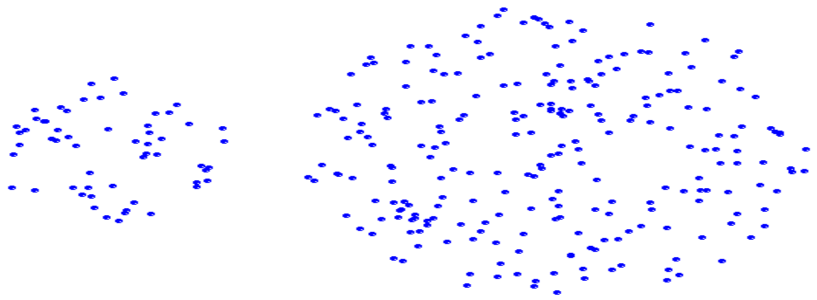


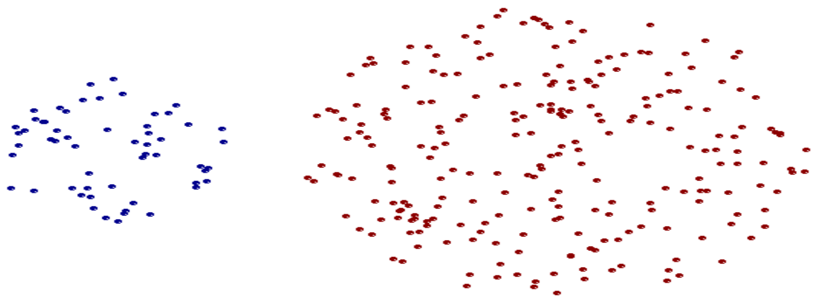


Nested Clusters

Dendrogram

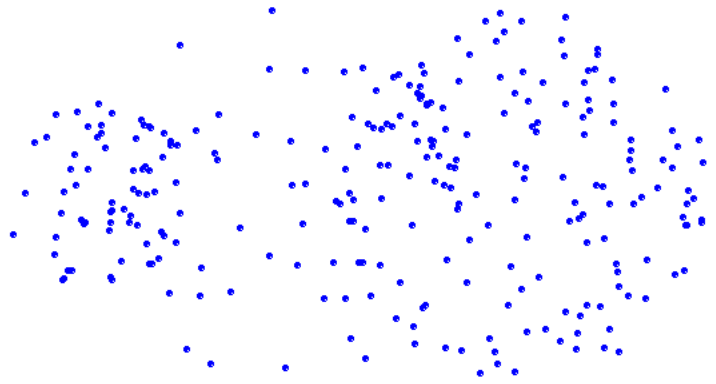
Strength of MIN


Original Points

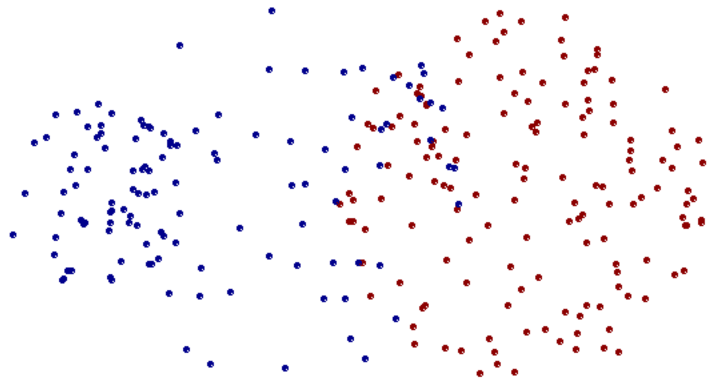

Two Clusters

- Can handle non-elliptical shapes

Limitations of MIN



Original Points



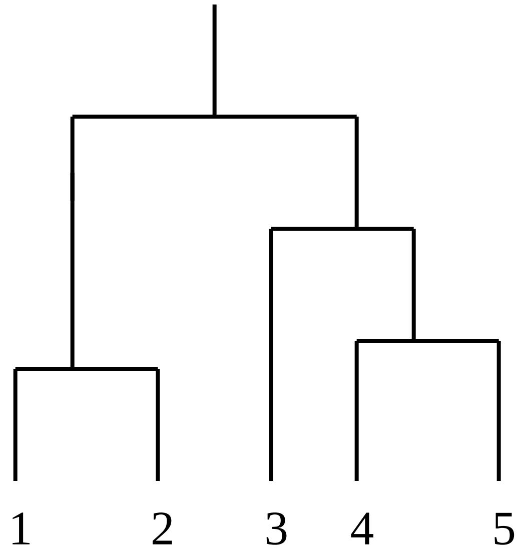
Two Clusters

- Sensitive to noise and outliers

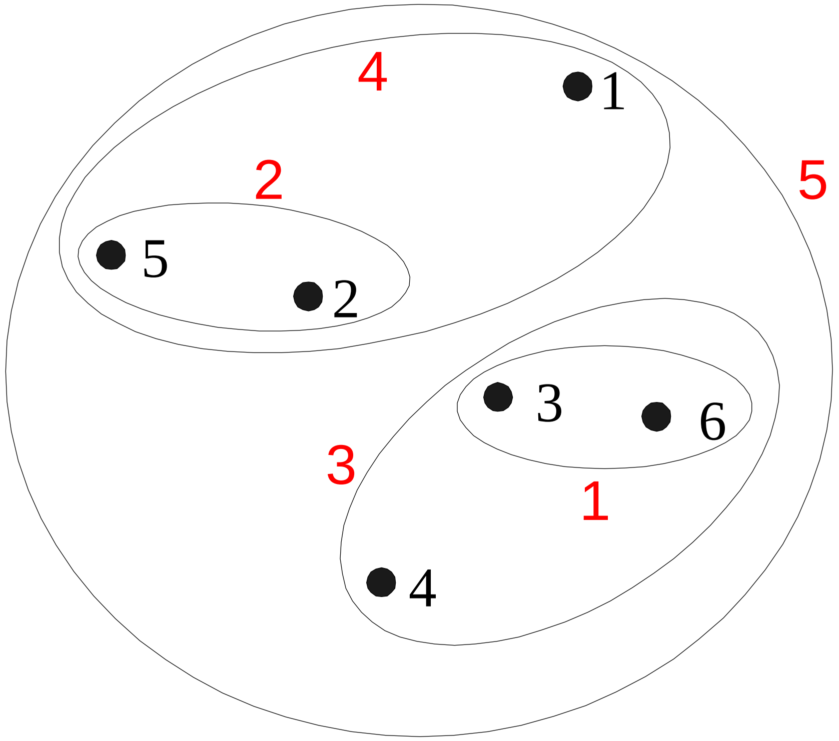
Cluster Similarity: MAX or Complete Linkage

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

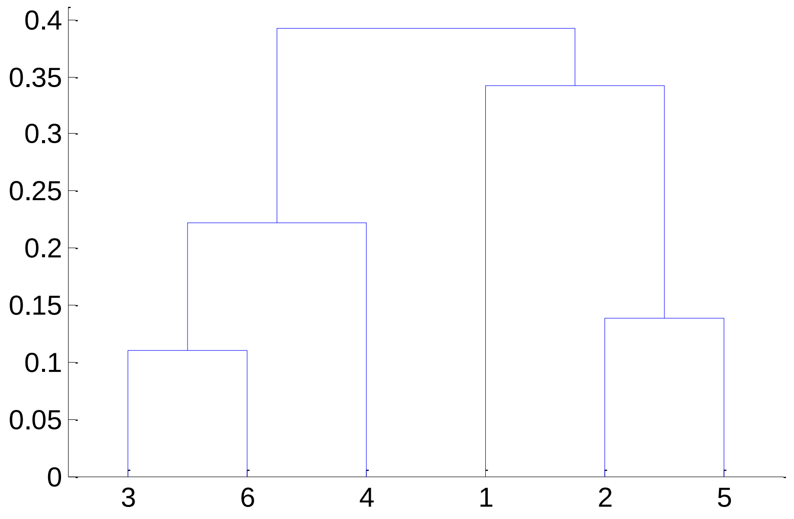
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MAX

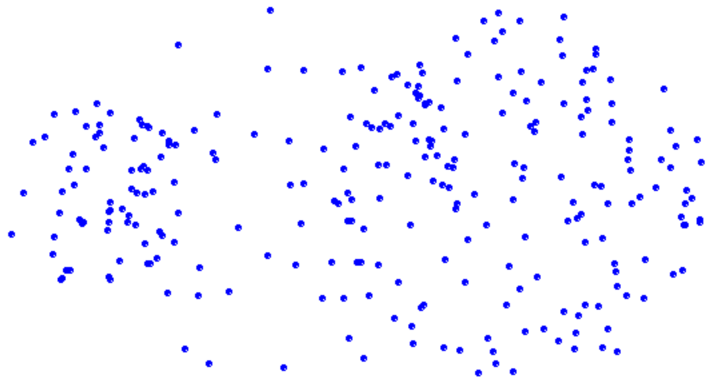


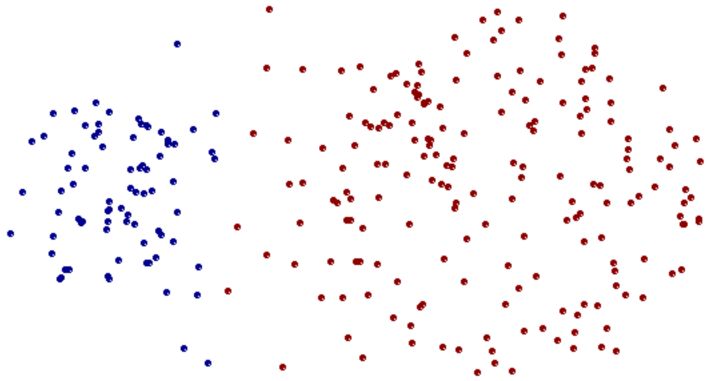
Nested Clusters



Dendrogram

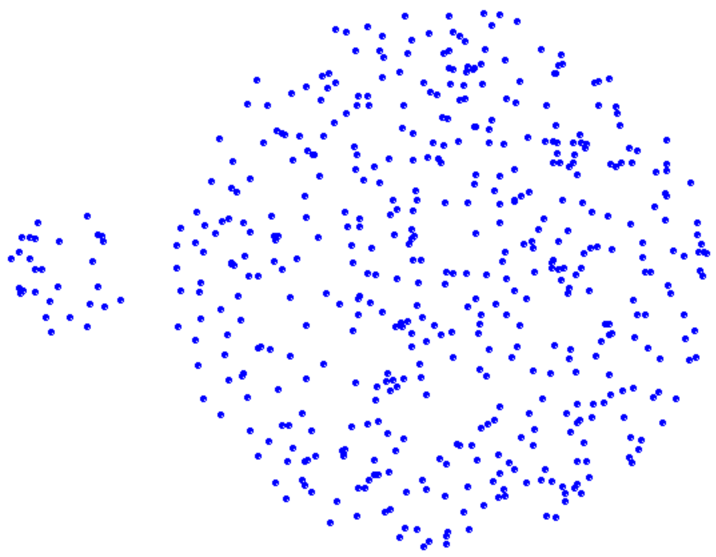
Strength of MAX

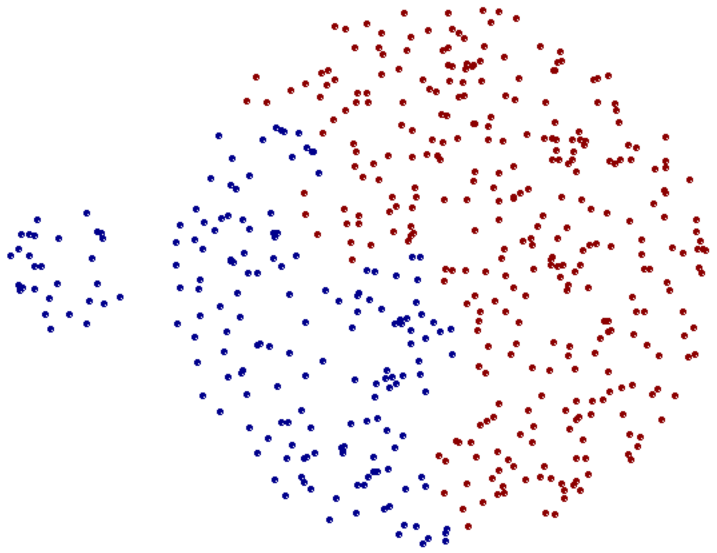

Original Points


Two Clusters

- Less susceptible to noise and outliers

Limitations of MAX


Original Points


Two Clusters

- •Tends to break large clusters
- •Biased towards globular clusters

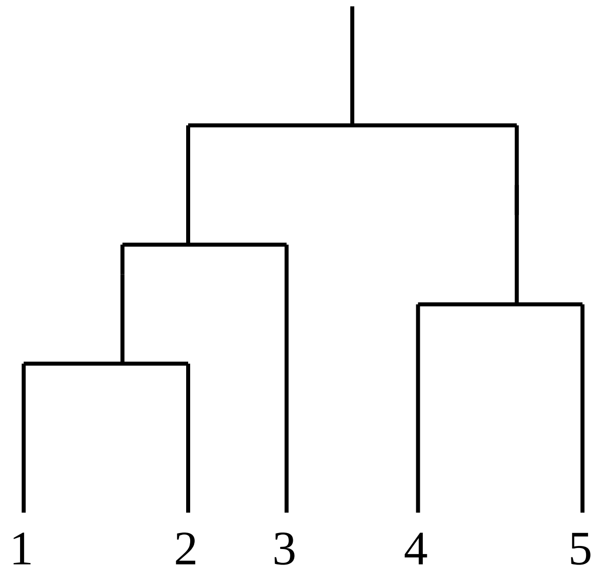
Cluster Similarity: Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

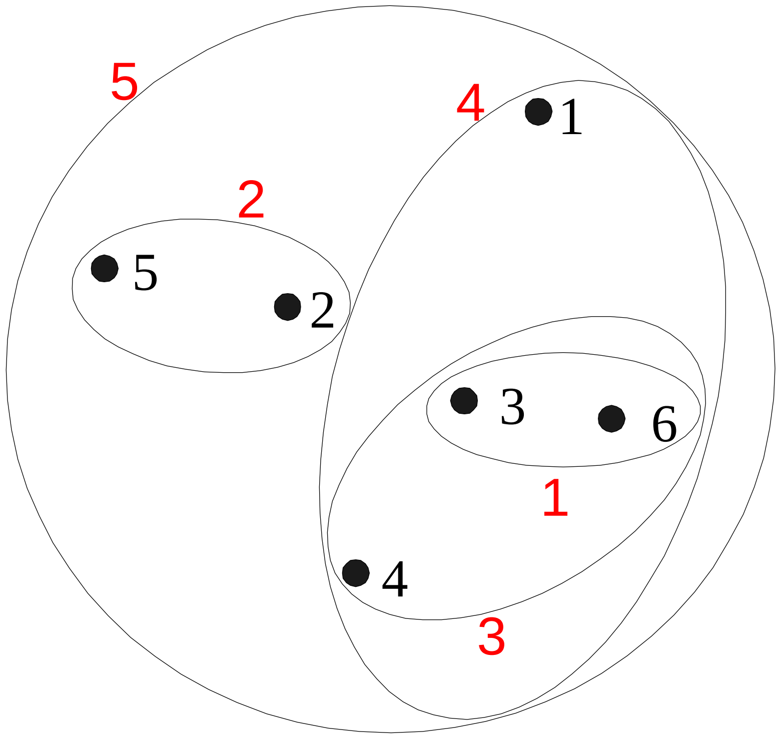
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum\limits_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

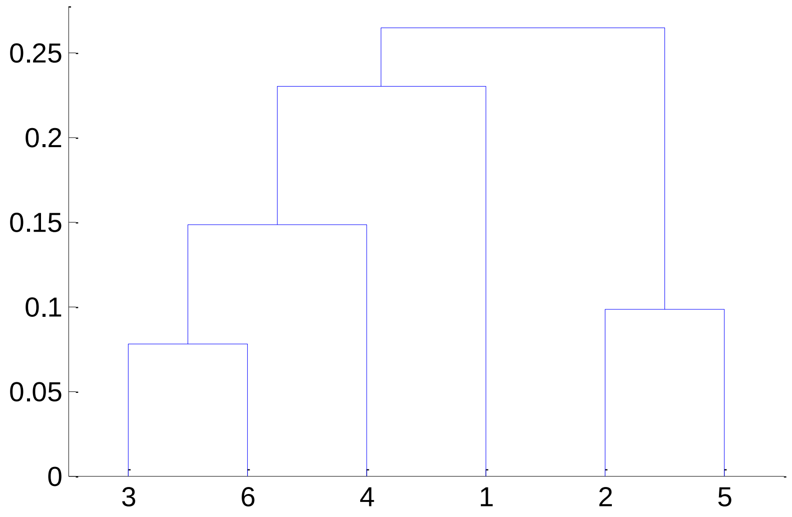
- Need to use average connectivity for scalability since total proximity favors large clusters

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: Group Average





Nested Clusters

Dendrogram

Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link

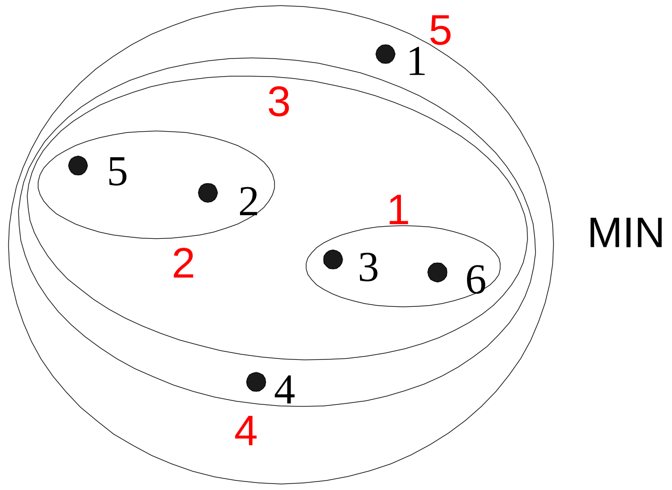
- Strengths
  - Less susceptible to noise and outliers

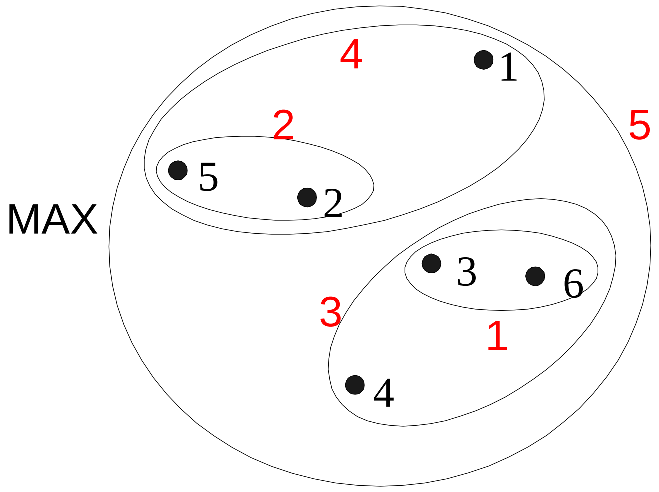
- Limitations
  - Biased towards globular clusters

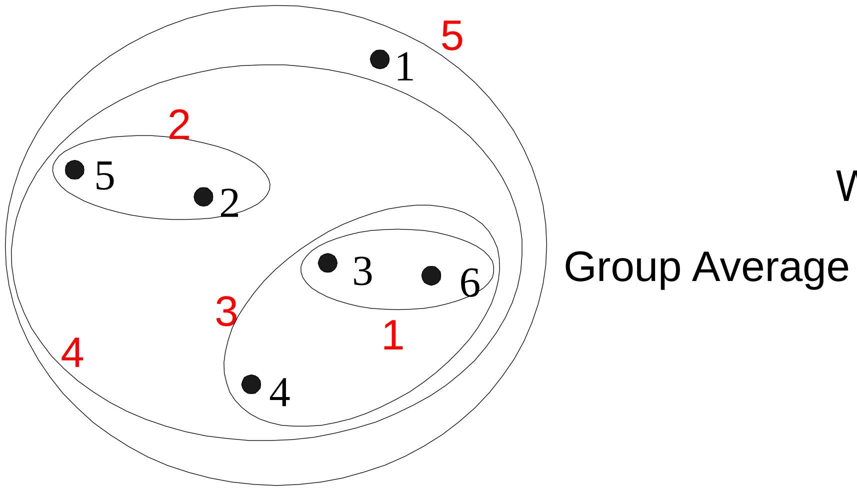
Cluster Similarity: Ward's Method

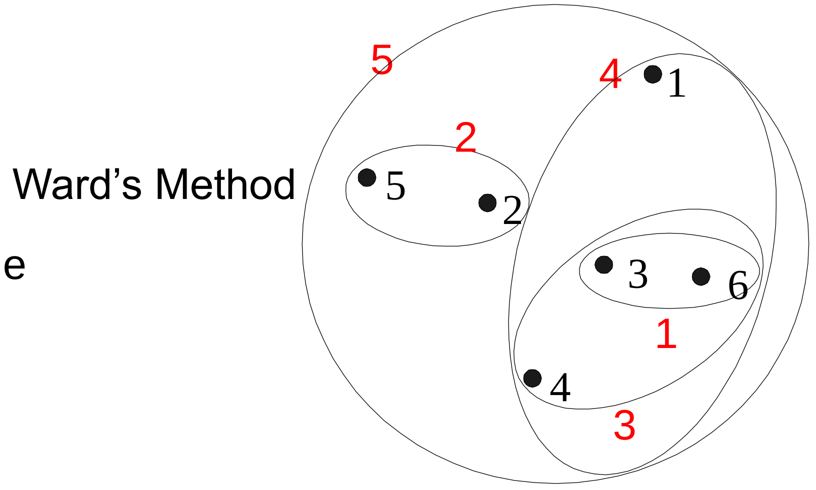
- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize K-means

Hierarchical Clustering: Comparison









Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.

- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters